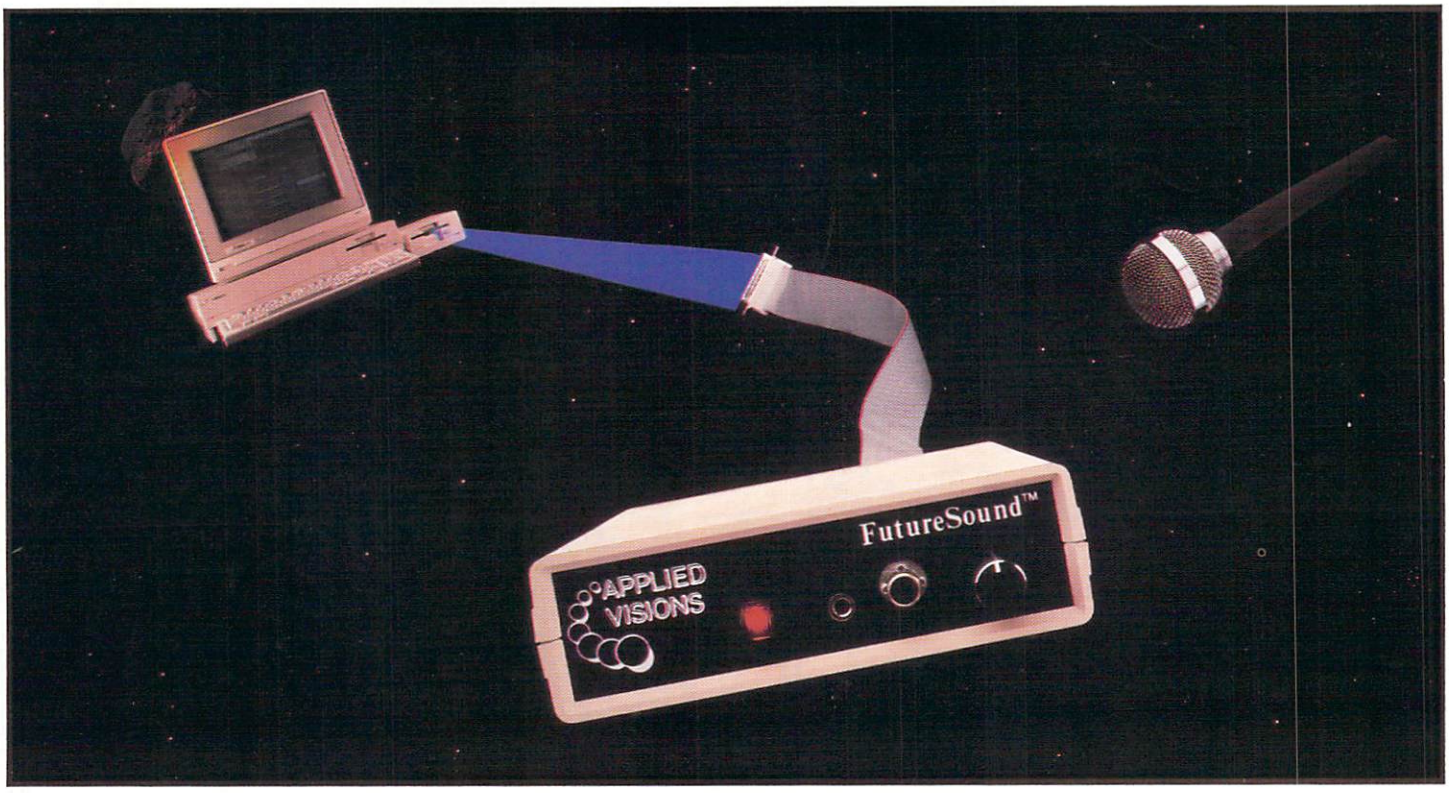
AMIGA Numbers Summed Up!

Amazing COMPUTING™

Your Original AMIGA™ Monthly Resource

Volume 2 Number 11
US $3.50 Canada $4.50

Graphics Update:
Digi-Paint & Sculpt 3D



"Open the pod bay doors, HAL..."

Programmers cast their vote!

Right now, leading software developers are hard at work on the next generation of Amiga® products. To add the spectacular sound effects we've all come to expect from Amiga software, they are overwhelmingly choosing one sound recording package... FutureSound. As one developer put it, "FutureSound should be standard equipment for the Amiga."

FutureSound the clear winner...

Why has FutureSound become the clear choice for digital sound sampling on the Amiga? The reason is obvious: a hardware design that has left nothing out. FutureSound includes two input sources, each with its own amplifier, one for a microphone and one for direct recording; input volume control; high speed 8-bit parallel interface, complete with an additional printer port; extra filters that take care of everything from background hiss to interference from the monitor; and of course, a microphone so that you can begin recording immediately.

What about software?

FutureSound transforms your Amiga into a powerful, multi-track recording studio. Of course, this innovative software package provides you with all the basic recording features you expect. But with FutureSound, this is just the beginning. A forty-page manual will guide you through such features as variable sampling rates, visual editing, mixing, special effects generation, and more. A major software publisher is soon to release a simulation with an engine roar that will rattle your teeth. This incredible reverberation effect was designed with FutureSound's software.

APPLIED VISIONS

Question: What can a 300 pound space creature do with these sounds?

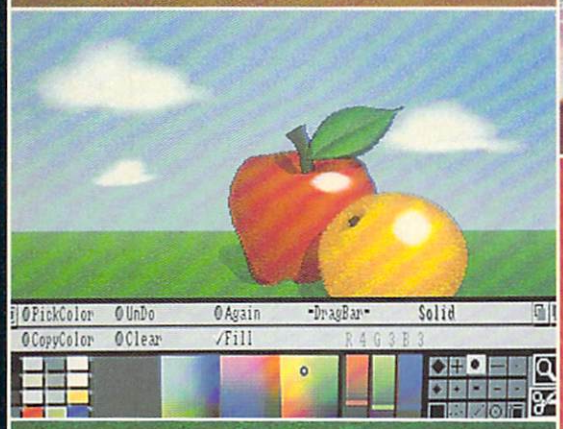
Answer: Anything he wants.

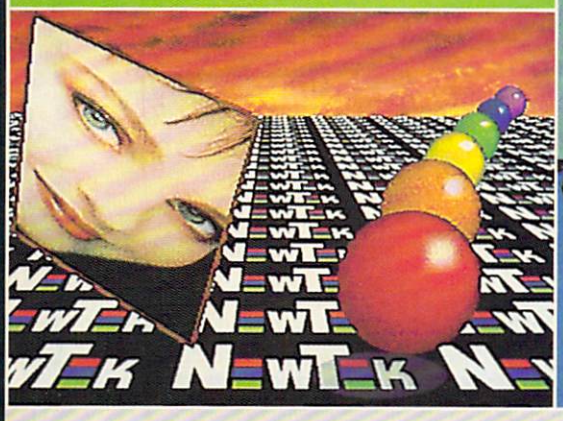
Since FutureSound is IFF compatible (actually three separate formats are supported) your sounds can be used by most Amiga sound applications. With FutureSound and Deluxe Video Construction Set from Electronic Arts, your video creations can use the voice of Mr. Spock, your mother-in-law, or a disturbed super computer.

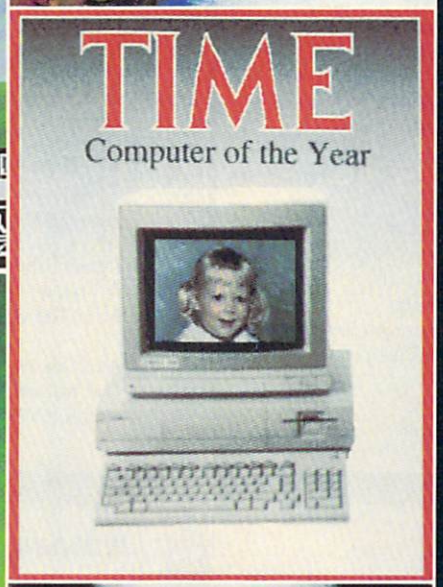
Programming support is also provided. Whether you're a "C" programming wiz or a Sunday afternoon BASIC hacker, all the routines you need are on the non-copy protected diskette.

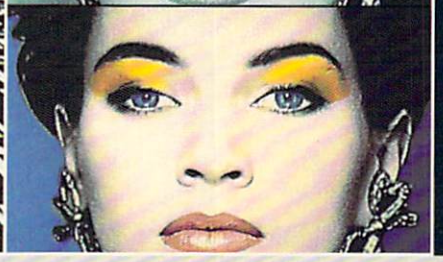
Your Amiga dealer should have FutureSound in stock. If not, just give us a call and for $175 (VISA, MasterCard or COD) we'll send one right out to you. Ahead warp factor one!

Applied Visions, Inc., Suite 2200, One Kendall Square
Cambridge, MA 02139 (617) 494-5417









ONLY DIGI PAINT CAN DO ALL THIS

Get the maximum graphics power from your Amiga. Create stunning, lifelike computer artwork with Digi-Paint, the first full-featured 4096 color (Hold and Modify) paint program. Break the "32 color barrier" and finally realize the potential of your Amiga with Digi-Paint's advanced features:

- 4096 colors on screen simultaneously
- NewTek's exclusive enhanced HAM mode
- Dithered HAM gradient fill
- Full screen effects including double, half size, mirror reverse and more
- Full IFF and Digi-View compatibility
- Use 320x200 or HAM hi-res 320x400 resolutions
- Fat bits Magnify mode
- Rectangle, oval, line and other drawing tools
- 12 different paint modes including blending, tinting and smooth shading
- Full lasso cut and paste with automatic edge blending
- Programmed completely in assembly language for fast, smooth response

Find out why Byte Magazine called Digi-Paint "Remarkable". Available now at your local Amiga dealer or call: **1-800-843-8934.**

ONLY $59.95

NEWTEK
INCORPORATED

Amazing COMPUTING™

Volume 2, Number 11

CONTENTS

Amazing Features

Word Processors Rundown by Geoff Gamble 19
ProWrite, Scribble! and WordPerfect face off on a writer's desk.

LPD Writer Review by Marion Deland 23
An Amiga word processor "you can just sit down and use."

VizaWrite Review by Harv Laser 29
A powerful, easy to use word processor with great Amiga font output.

Aedit Review by Warren Block 36
Put your text-editing frustrations to rest with this reliable, full-featured editor.

WordPerfect Preview by Harv Laser 41
The top selling IBM PC word processor makes a successful transition to the Amiga.

Jez San Interview by Ed Berovitz 44
Candid comments from the StarGlider author on his background, current projects and the Amiga in Europe.

Do-it-yourself Improvements to the Amiga Genlock by John Foust 88

Amazing Reviews

Digi-Paint Review by Harv Laser 8
"A new generation paint program that operates in the 4096 color HAM mode . . ."

Sculpt 3D Review by Steve Pietrowicz 14
Now you can design realistic 3-D images on your Amiga.

Shadowgate Review by Linda Kaplan 49
Adventure is abounding in this game of stunning special effects and graphics.

TeleGames Review by Michael T. Cabral 71
Amiga gaming steps into another dimension . . . strategy via modem.

Reason Preview 63
A preview of a multi-faceted proof-reading package from The Other Guys.

As I See It by Eddie Churchill 43
Our newest column takes a peek at WordPerfect, Gizmoz V2.0 and Zing! Keys

Amazing Programming

C Animation Part II by Mike Swinger 56
The "Animation for C Rookies" series continues with a look at Animation Objects.

BASIC Text by Bryan Catley 66
Tips for positioning text to specific pixel locations that don't match row/column locations.

Soundscape Part III by Todor Fay 75
The Soundscape author helps you create a module not displayed in the Patch Panel . . . aVU Meter.

Fun With Amiga Numbers by Alan Barnett 83
" Uncover the secrets of floating point arithmetic within your Amiga."

File Browser by Bryan Catley 93
A full-featured file browsing utility that you can program.

Amazing Columns

Bug Bytes by John Steiner 55
Keep up with all the latest bugs and upgrades.

AmigaNotes by Rick Rae 60
A look inside four electronic music books.

Modula-2 Programming by Steve Faiwiszewski 103
The series rolls along with devices, I/O andthe serial port.

Roomers by the Bandito 101
The rumor mill continues to spill over.

68000 Assembly Language by Chris Martin 111
Tune in as Chris walks us through the display routines.

The AMICUS Network by John Foust 116
Desktop publishing and a report from the Seybold conference.

Amazing Departments

Amazing Mail 4
Index of Advertisers 112
PDS Software Catalog 123

Amazing COMPUTING™

Publisher:	*Joyce Hicks*
Circulation Manager:	*Doris Gamble*
Asst. to the Publisher:	*Robert James Hicks*
Corporate Trainer:	*Virginia Terry Hicks*
Traffic Manager:	*Robert Gamble*
Managing Editor:	*Don Hicks*
Submissions Editor:	*Ernest P. Viveiros Jr.*
Hardware Editor:	*Ernest P. Viveiros Sr.*
Music & Sound Editor:	*Richard Rae*
Art Director:	*Keith M. Conforti*
Advertising Manager:	*John D. Fastino*
Copy Editor:	*Michael T. Cabral*
Amicus & PDS Editor:	*John Foust*
Production Manager:	*Mark Thibault*
Assistant PM:	*Keven P. Desmarais*

Advertising Sales & Editorial
1-617-678-4200

Special thanks to:
Lynn Hathaway
Donna Peladeau
Traci Desmarais
Pilar Medeiros
Donna Thibault
Kristen Simbro
Betsy Piper at Tech Plus.
& Paul Boden at Software Supermarket

Amazing Computing™ (ISSN 0886-9480) is published by PiM Publications, Inc., P.O. Box 869, Fall River, MA 02722. Subscriptions: in the U.S.12 issues for $24.00; in Canada & Mexico, $30.00; Overseas; $35.00. Printed in the U.S.A.

First Class or Air Mail rates available upon request.

PiM Publications, Inc. maintains the right to refuse any advertising.

PiM Publications, Inc. is not obligated to return unsolicited materials. All materials requesting return must be received with a Self Addressed Stamped Mailer.

Amazing Mail:

Dear Amazing,
I just received Vol 2.9 (another excellent production, by the way) only to find extremely disconcerting news in the editorial. I can't believe that anyone would feel the need to pull such a low level stunt as to remove the copyright text from a program and post it as PD. Not only is Inovatronics harmed (their products are quite good), but if this incident leads to even more elaborate copy protection industry wide, then we all lose. This comes at a time when it seems that many software houses were just beginning to warm up to the idea of non-disk based copy protection.

I can only hope that Fred Fish is not held liable for legal action. Not only would this be an injustice to an extremely reputable man, but it would in no way discourage software piracy. Please keep us informed about any legal action thay may come about as a result of this. I'm quite sure that a vigorous write in campaign could be staged for the benefit of Mr. Fish, if needed.

Thank you,
Charlie Mattax
Miami Lakes , FL

Dear Amazing,
I support Fred Fish's effort to provide a public domain software library that is good. I hope the company who had its software pirated recognizes Fred's effort to honestly promote the Amiga and will support Fred by not bringing any legal action against him. Keep up the good work Fred!!

Pauls S Lubertowicz
Mansfield, PA

Dear Gentlepeople,
Your editorial on pirating software is on the mark. Pirates steal from all of us by making software more expensive and having to cope with the copy protection that folks like Electronic Arts seem to think necessary.

Even worse, pirates limit what will be on the market. By stealing a part of a product's sales, they lessen the developer's profit livelihood and incentive to develop additional marketable programs.

An example of this was related to our user group by a developer. In answer to an inquiry about the potential market for Amiga software in Germany, a distributor told him a "Hit" sold about 50 copies - 50 copies for about 50,000 Amigas. That Germans write their own programs is not the likely reason, because Germany is not a hotbed of Amiga software. And why - because the pirates have destroyed the incentive.

So much for the soapboxing and now for THE NEWS. The Amiga Group, a College Park (U MD) user group has completed their expansion box project and are offering it as a kit to local user group members. The box has 5 expansion slots and a 200 watt power supply and should cost less than $300.00, depending on how many amigos join in on the project. The first, and probably only production run will be in October, then TAG will leave the field open to folks like ASDG and their 2000 and 1 box. Joe Kaisler has honcho'd the project; he can be reached at: 12034 Beltsville Dr; Beltsville,MD 20790; or the TAG BBS (301) 572-2471.

Amazed
Carl S. Custer
Bethesda,MD 20817

Dear Mr. Hicks,
We received V2.9 of Amazing Computing last week and we read your editorial on "Piracy and Undue Harm." We would like to take this opportunity to clear up a couple of things.

Fred Fish is not "awaiting notification of possible criminal charges" because no such threat was ever made. At the time, we considered CIVIL litigation, but criminal proceedings were never mentioned. Since we have decided against litigation, we won't be taking any action at all against Mr. Fish. Upon reading the editorial (and the sidebar on page 94), we sent Mr. Fish a letter confirming the fact that we were not taking any kind of action against him and were officially closing the matter.

The reasons why we considered legal action against Fred should be obvious. The four-program package "M," developed in-house and still copyrighted by Inovatronics, retails (retailed, I should say now) for $59.95. Fred distributed it widely, if unintentionally, into the public domain. The package is not particularly impressive by today's Amiga software standards (M was released circa Feb. 86), and was not selling very well at the time Fred released it. Loss of commercial value is not the real issue in this one case.

Our number one product, PowerWindows, has turned up on several "pirate" BBSs. We are in no way likening Fred Fish to these real pirates. However, we really do not want to establish a precedent of simply allowing the free distribution of our proprietary software, or anyone else's for that matter. We develop and sell software for a living. We must retain our right of ownership in order to survive and, by writing software, help the Amiga do the same.

Fred Fish is, of course, invaluable to the Amiga and Amiga owners. We would like very much to see the non-copyrighted programs on Fish disks 80 and 88 reappear in the collection. We don't want to be thought of as the company that caused the "holes" in his fine library.

We hear that Fred is thinking about requiring either source code or proof of authorship for future programs going into his collection. We feel this will help avoid misunderstandings like the one in which we became involved.

We all want a bright future for the Amiga.

Sincerely,
Tom Hardison
Promotions Director
Inovatronics, Inc.

Amazing Dealers

The following are Amazing Dealers, dedicated to supporting the Commodore-Amiga™.
They carry Amazing Computing™, your resource for information on the Amiga™.

If you are not an Amazing Dealer, but would like to become one, contact:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA. 02722
1-617-678-4200

Alabama
Abax Data Systems — Huntsville
Alabama Book Store — Tuscaloosa
Command Computers — Birmingham
Electronics Boutique — Hoover
Madison Books & Computers — Madison
Mail Tel Communications — Mobile
Mel's Photo & Computer Shop — Montgomery
Universal Computer System — Mobile

Arkansas
Computermart — Mountain Home
SIS Inc. — Mt. Home
The Micro Shop — Little Rock
The 64 Store — Fayetteville
The 64 Store — Jonesboro
The 64 Store — North Little Rock

Arizona
Computer Library, Inc. — Tempe
Computer West — Phoenix
Computer Works — Glendale
Copperstate Business Systems — Phoenix
Hacko's — Tempe
North American Digital — Tucson

California
Adag International — Foster City
Augnet — Fremont
Balboa Office Supply — San Diego
Bits n Pieces Comp Prod. — Anaheim
Brown Knows Computing — Redlands
Byte Factory Computers — Salinas
Century Computer Systems — La Habra
Circus of Books — Los Angeles
Circus of Books — Los Angeles
Commotech — National City
Compunik — Los Angeles
Computer Attic — Palo Alto
Computer Cellar — Anaheim
Computer Nook, Inc. — San Bernadino
Computer Outlet — San Diego
Computer Outlet — San Marcus
Computer Time — Citrus Hts
ComputerLand Fremont — Fremont
Coronet Newsstand — Oceanside
Dataphile — Watsonville
Delta Spectrum — Laguna Beach
Empire Computers — Santa Rosa
ECX Computer Co. — Walnut Creek
Harding Way News — Stockton
Heinrichson, Schneider & Young, Inc. — Berkeley
Home Computing Center — Concord
Home Computing Center — Mill Valley
Home Computing Center — San Leandro
Home Computing Center — San Bruno
HT Electronics — Sunnyvale
K J Computers — Granada Hills
Lewty — North Hollywood
LTC Computers — Santa Ana
Micro Flight — San Jacinto
MTS Inc. Tower Books — Sacramento
Nonagon Software — El Cerrito
Op Amp Tech Books — Los Angeles
Paras Newsstand — San Diego
Pat's Electronics — Rio Linda
R&R Software — Chula Vista
Real Computers & Electronics — Petaluma
Ridgecrest Computer Ctr. Inc. — Ridgecrest
Sawyer's News Inc. — Santa Rosa
S.O.S. Computers — Los Angeles
Software First — Santa Rosa
Software Cellar — Anaheim
Software Central — Pasadena
Software Plus — Citrus Heights
The Booksmith — San Francisco
Tower Books — Chico
Tower Books — Concord
Tower Books — Mountain View
Tower Books — Sacramento
Tower Books — Sacramento
Tower Books — Sacramento
Tower Books — San Mateo
The Computer Room — Scotts Valley
Winner's Circle — Berkley
World Book & News — Hollywood

Colorado
Aurora Newsstand — Aurora
Book Tech Distributing — Englewood
Citadel Computers — Colorado Springs
Computer Discount — Denver
Computer Room — Aurora
Ebert Personal Computers — Aurora
Micro Computer World — Fort Collins
Micro World — Lakewood
Sun Country Computers — Colorado Springs

Connecticut
Aetna Life Club Store — Hartford
Amiga Business Computer — Norwalk
Computer Outlet — Old Saybrook
Friendly Computers — Plantsville
Mnemonics Inc. — Norwalk
Personal Computer Center — Norwich
Software City — Orange
Software Kingdom — East Windsor
Spectrum Computers — New Haven
Tricom Computers — North Windham

Delaware
Castle Video — Newark

Florida
A A Computers — Jacksonville
Commodore Country Inc. — Pinellas Park
Computer Bar — Pennsacola
Computer Base — Brandon
Computer Image — Miami
Computer Specialty — Melbourne
Computer Terminal Inc. — Sebring
Computers 4 Rent — W. Palm Beach
Computers Plus — S. Daytona
Cotronics Inc. — Stewart
Data Base — Davie
Education Computers Etc. — Tallahasee
Family Computers — Sarasota
Florida Book Store — Gainesville
Gulf Coast Computer Exchange — Panama City
Heartland Computers — Mulberry
Joe's News, Inc. — Miami
K&L Computers — E. Palatka
Megaport Computer Center — Venice
Micro Computer Center — Largo
New Age Electronics — St. Petersburg
Program Store — Clearwater
Random Access Computers Inc. — Ft. Walton Beach
Software City — Orange Park
Software Software — Plantation
Sunny's At Sunrise, Inc. — Sunrise
The Open Door — Cocoa
The PC Collection — Coral Gables

Georgia
Future System — Augusta
Software South Inc. — Savannah
The 64 Store — Atlanta

Hawaii
Astro Computers & Electronics — Aiea
Computer House Inc. — Honolulu
J & C Repair — Aiea

Idaho
A.B.I. Computer & Video — Nampa
Boise Computer & Video Inc. — Boise

Illinois
A Bitchster — Rockford
Computerland O'Hare — Park Ridge
ComputerLand of Niles — Niles
DigitalWorld — Addison
I.C.V. Data Center Inc. — Mundelein
Illini Microcomputers Inc. — Naperville
Interstate Computer Service — LeMont
Krach's & Bertano's — Chicago
Micro Age — Oaklawn
Software Of Systems — Fairview Hts.
Software Plus — Wheeling
Software Plus West — Hanover Park
The Memory Expansion — Normal
Unique Computer — Sterling

Indiana
Burkat Computer Ctr. — South Bend
Byrex of Angola — Angola
Computer Corner — Fort Wayne
Computer Corner — Marion
Computer People Inc. — Michigan City
Greg Chaney — Greenfield
Indiana News — Indianapolis
Micro Computer Centers Inc. — Indianapolis
Spectra Computer Sys. Inc. — Evansville
Vons Computers — West Lafayette

Iowa
Gronet Computers — Des Moine
Micro Computer Applications LTD — Marshalltown

Kansas
J & L Electronics — Liberal
MicKansas Computers — Newton
Midwest Computers — Manhattan
Software City — Overland Park

Kentucky
Computer Dimensions Inc. — Lexington
Computer Store of Kentucky — Louisville
Hawley Cooke — Louisville
Reality World Computers — Lexington

Louisiana
Computer Time — Shreveport
Modern Business Machines — Metairie
Software Center International — Metairie
Software Solutions — Baton Rouge
Software Source — Lafayette
The Bok Shelf — Slidell
The Computer Clinic — Lafayette

Maryland
Buried Treasure — Rockville
Computerware TODAY — Gaithersburg
Compuvision Vision Center — Columbia
Layhil Newsstand — Kensington
Micro Computer Center — Baltimore
Software Advantage — Rockville
Software N Things — College Park
Waldorf Computer — Waldorf

Massachusetts
Club Computer — Cambridge
D-Five — Bedford
DOX — Williamstown
General Computer Store — Framingham
HCS Computer Center — Marshfield
LCA Video & Computer Ctr. — Norwood
Memory Location — Wellsley
OmniTek Computers — Tewksbury
Pioneer Valley Data Equipment — Amherst
Softpro — Burlington
The Bit Bucket — West Newton
The Software Haus — Quincy
Tycom Inc. — Pittsfield

Maine
Computer Barn — Damariscotta
Hands On Computing — Westbrook
Radio Shack Associate Stores — Jay

Michigan
Basic Computer Center — Grand Rapids
Bits-Bytes-Nibbles — Petowskey
Canton Computer — Canton
CompuSoft — Kalamazoo
Direct Access — Lansing
Edmurton Computer Center — Pontiac
Global Computer Center — Dearborn
I O Software — Flint
Central Park Computers — Okemos
Michigan Software — Novi
Midland Computer Shop — Midland
Pro-Video — Okemos
Programs Unlimited — Grand Rapids
Readmore Newscenter — Lansing
Roseville Computer Store — Warren
Slipped Disk — Madison Heights
Specialists Computer — Livonia
State Street Computers — Ann Arbor
T.J. Enterprises — Clinton
The Edge Connector — Clio
The Soft Warehouse — Kalamazoo
Ye Olde Computer Shoppe — Ypsilanti

Minnesota
Computers, etc. — Eagan
Computer Place — Minneapolis
JMH Software — Maple Grove
MCD of Hibbing Inc. — Hibbing
Midwest Computer Services — Clearwater
Shinder's Read More — Minneapolis
Specialists IN — Hopkins
Valiant Inc. — Stillwater

Mississipi
National Computer Center — Ocean Springs

Missouri
Associated Computer Services — Springfield
Brands Mart Computers — Kansas City
Data Byte Computer Co. — St. Louis
Instant Replay Ltd. — Chesterfield
Systems Plus — Independence
Systems Plus of St. Louis — St. Louis

Montana
Appegren Computer Systems — Great Falls

Nebraska
BULLS-EYE — Lincoln
My Com Inc. — Plattsmouth
Double E Electronics — Omaha

Nevada
Century 23 Inc. — Las Vegas
Computer World — Las Vegas
Tower Records — Las Vegas

New Hampshire
Computer Mart of New Hampshire — Nashua
Diversified Computers — Keene

New Jersey
Family Computer Centres — South Orange
Morris County Stationers — Chester

New Mexico
New Horizon Computer System — Alamogordo
Page One Newsstand — Albuquerque
Technological Concepts — Las Cruces

New York
Amuse — New York
Amicus- Hudson Valley — Wappingers Falls
Amiga Business Computer — North Port
Byte Shop — Merrick
Compurite Center — White Plains
Computer Outlet — Jamestown
Computer Palace — Felden
Computers Etc. — Syracuse
Consumers Software — Commack
Cromland Inc. — Brooklyn
Leigh's Computers — New York
Mnemonics — North Port
N.U.B.S. Computer Center — Bronx
Software Centre International — Forest Hills
Software & Such — Scotia
Software City — Forest Hills
Software Supermarket — Kenmore
Star Tech Systems — Massena
Syntaxe Trading — New York
Tadiran — New York
Tedrow Business Products — Rochester
The Computer Cellar — Albany
The Working Computer — Stoney Brook
Video Computer Bin — Brooklyn
Video Computer Center — Rome
Video Connection — Schenectady
Village Computer Inc. — New York
World Computers Inc. — Hicksville

North Carolina
C2 of Hickory — Hickory
Digitz — Raleigh
K&S Newsstand — Clemmons
K&S Newsstand — Kernersville
K&S Newsstand — Winston-Salem
K&S Newsstand — Winston-Salem
K&S Newsstand — Winston-Salem
Software City — Raleigh
Software City — Charlotte
TDS Computers — Durham
Triad Computers — Winston-Salem

North Dakota
Computer Associates — Fargo
G. F. Computers — Grand Forks
The Computer Store — Minot

Ohio
Computer Network — Newark
Computer Site — Cleveland
Earthrise Micro Systems — Columbus
Fairborn Home Computer — Dayton
Lakes Consumer Electronics — Akron
Main Office Supply — Coshocton
Micro Center — Columbus
Microwave Magic — Fairfield
North Coast Programming — Willoughby
Personal Computer Co. — Youngstown
Paper & Ink — Tiffin
Quality Computer Application — Toledo
Saxton Computer Center — Sandusky
Softsales Centre International — North Olmsted
Software and More — Cincinnati
Software City — Centerville
Software Connection — Mayfield Hts.

Oklahoma
Colonial Video & Computer — Bartlesville
Micro Link Computer Store — Tulsa
P C Tech — Tulsa
Second Hand Software — Oklahoma City
Video-Comp Inc. — Lawton

Oregon
Andromeda — LaGrande
Capitol News Center — Salem
Clackamas Computers — Beaverton
Clackamas Computers — Clackamas
Computron Business Systems — Portland
Powell's Books — Portland
Powell's Tech — Portland
Rich Cigar Store — Portland
Software Express — Eugene

Pennsylvania
Alpha Omega EBE — Warren
Basic Computer Systems — Hermitage
Chester County Book Co. — West Chester
Circle Computer — Ephrata
Computers and Games — Reading
Downington Computer Center — Downington
East Coast Software — Hershey
Floppy Disk — Mechanicsburg
Gene's Books — King of Prussia
North Central Digital Systems — Danville
Pittsburgh Computer Store — Pittsburgh
Triangle Computers — Indiana

Rhode Island
Micro Limits — Smithfield
Micro Limits — Warwick
Software Connection — Warwick

South Carolina
Avcom International Inc. — West Columbia
Horizon — Greenville
Micro Computer Depot — Sumter
Softwares City — Spartanburg
Watson's Jewlery — Chesterfield

South Dakota
Computer Business Center — Rapid City

Tennessee
ComputerLab of Memphis — Memphis
The 64 Store — Bartlett

Texas
Bookstop — Houston
Byte Shop — Austin
C.E.M. Corporation — Brazoria
Colonel Video — Houston
Computer Age — Houston
Computer Age — Dallas
Computer Magic — Austin
FNS Computer Services — Texarkana
Metropolitan Computer — Richardson
Micro Search — Houston
Software Library — Wichita Falls
Software Terminal — Fort Worth
The Computer Experience — San Antonio
The Computer Stop — Abilene

Utah
Computers Plus — Murray
Magazines and Gifts — Salt Lake City
People's Computers — Provo

Vermont
Ray Supply Inc. — South Burlington

Virginia
Alonso — Alexandria
Diskovery — Falls Church
Family Computer Center — Fairfax
I/O Computer — Norfolk
Software Central — Norfolk
Software City — Richmond
Virginia Micro Systems — Woodbridge

Washington
A.P.P.L.E. CO-OP — Renton
Apple Valley Computers — E. Wenatchee
Bits, Bytes and Nibbles — Spokane
Bull Dog News — Seattle
Byte World — Kent
Com-Soft Computers — Everett
Computer Exchange Services — Bellingham
Computer Quest — Edmonds
Computers + Inc. — Tacoma
Computers and Movie Magic — Yakima
Family Computers — Lynwood
KAOS — Northeast Poulsbo
Nybbles & Bytes — Tacoma
Omni International — Seattle
Programs Plus — Seattle
Real Time Systems — Seattle
Take A Byte — Kennewick

Wisconsin
Colortron Computers — Racine
Computer Software Center — Milwaukee
Fox Valley Personal Computer — Oshkosh
Mom's Computers — Stevens Point
Richard Reske — Madison
TMW Software Inc. — Wausau

Wyoming
Byte Emporium — Laramie

Canada
Compute or Play — Alberta
Computerworks — Alberta
Computer Shop — Alberta
Data Pro — Alberta
Computer Shop of Calgary — Alberta
The Software House — Alberta
Hindsin Computing — Alberta
Red Deer Micro Systems — Alberta
Memory Land — Alberta
Computer King — British Columbia
Coms — British Columbia
Nutec — British Columbia
Sprite — British Columbia
Strider — British Columbia
Super Software — British Columbia
Canadian Computer Era — Manitoba
Micromart — Manitoba
ComputerWorld — Manitoba
J.W. Allen — Manitoba
Kobetek — Manitoba
Minicomp — Manitoba
Arkon — Ontario
Chatam Computer — Ontario
Computerland — Ontario
Comspec — Ontario
Electronics 2001 — Ontario
Hastings Data — Ontario
The Computer & You — Ontario
Thornhill — Ontario
Software Co. — Quebec
Deltrone Solare — Quebec
Mierodata — Quebec
Info Mercier — Quebec
Laval Micro — Quebec
Vision Tronique — Quebec
Distribution Valpro — Quebec

Distributed in Canada by
Ingram Software Distributors — Concord, Ontario
Compulit Distributors — British Columbia
Phase 4 Dist. Inc. — Calgary, Alberta

Argentina
Logic Computer — Buenos Aires

Australia
Diskworks — Greenwich, NSW
High Technology Computer Supplies — Brighton, Melborn
John Vernovan — Cloverdale, WA
Maxwell Office Equipment — Abbotsford, Melburn
Megatronics — Lanecove,NSW

Saudia Arabia
Abdulla Fouad & Sons — Daman

Scotland
The Amiga Center of Scotland — Edinborough

Switzerland
Software Land AG — Zuerich

Amazing Computing™ is also available in most B. Dalton Booksellers stores, B. Dalton Software stores, Crown Books, and Software Etc. locations.

Thanks to everyone who responded to our editorial. As Mr. Hardison stated, Inovatronics has dropped all pretense of seeking restoration through Mr. Fish.

However, this does not end the problem. Software piracy has kept many good developers from producing material on piracy Infested machines. This is why there is so little support for these machines. The Amiga has a good reputation. This one black mark should not turn away prospective developers, but it should caution all Amiga owners to safegaurd the Amiga's reputation.

Dear AC,
Having recently acquired several disks full of public domain software, I decided to use the Disk Librarian program that appeared in Amazing Computing™ Vol 2.2 to generate a program listing. The program worked well with most disks, however about 10% of the disks would cause the program to bomb. I traced the problem to FILENOTEs, which had been attached to some of the files on each of the disks.

To prevent the program from reading FILENOTEs, the following change seems to work well:

In the "adddisk" subroutine, change the following line from:

```
IF (INSTR(temp$,".info")=0= THEN
```

to:

```
IF ((INSTR(temp$,".info)=0) AND
(LEFT$(temp$,1)<>CHR$(58)))
THEN
```

FILENOTEs are prefaced with a colon. This addition will check for the colon and bypass the line if it finds one. Using the INSTR command does not work because the program finds the colons associated with the time stamp on each file, causing it to ignore every file. Also, using ":", instead of CHR$(58) generates a "WHILE without END" error, but I have been unable to determine why this occurs.

This change should not affect the program's ability to locate legitimate files because the operating system does not accept the use of a colon within a filename.

Regards
John W Quarterman
Plano,TX

As always, thanks for the help.

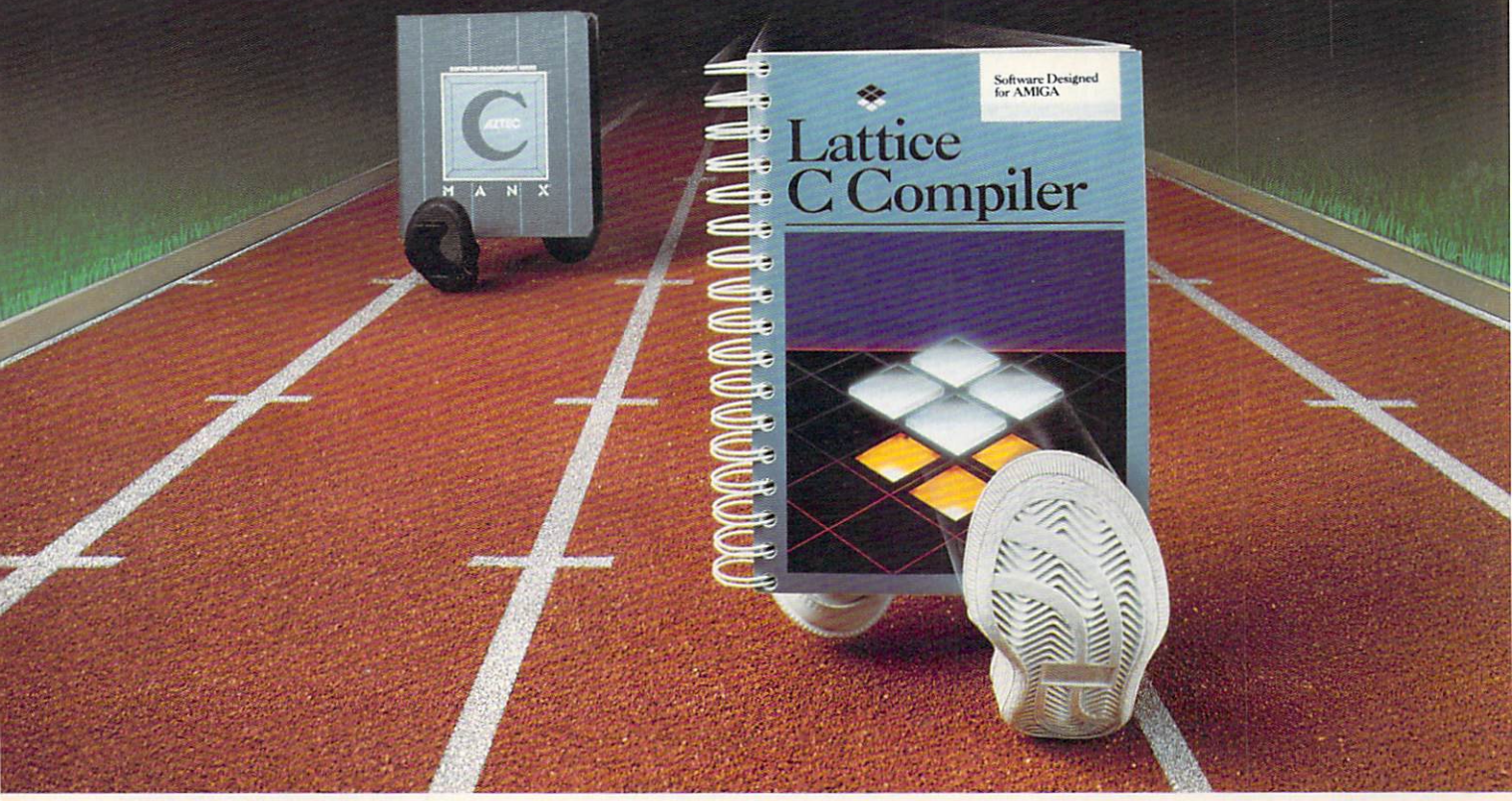
Dear Amazing Computing™:
In a recent publication of Amazing Computing™, credit was given to Chris Crawford as the designer of THE BARD'S TALE. This is incorrect. The designer is Mr. Brian Fargo. We appreciate the correction being made.

Sincerely,
David Dempsey
Public Relations Coordinator
Electronic Arts

Correction noted, thanks. Our apologies to Mr. Fargo. His work is very much appreciated..

•AC•

C who's winning the race. Lattice C for Amiga.



Lattice C has long been recognized as the best C compiler. And now our new version 4.0 for Amiga™ increases our lead past the competition even further.

Ready, set, go. The new Lattice AmigaDOS C Compiler gives you faster, more efficient code generation and support for 16 or 32-bit integers. There's direct, in-line interface to all Amiga ROM functions with parameters passed in registers. What's more, the assembler is fully compatible with Amiga assembler syntax.

More great strides. The linker, Blink, has been significantly enhanced and provides true overlay support and interactive recovery from undefined symbols. And you'll have a faster compile and link cycle with support for pre-linking.

	Lattice® Version 4.0	Manx® Version 3.40
Dhrystone	1294 Dhrystones/second	1010 Dhrystones/second
Float	22.20 Secs. (IEEE Format) 10.16 Secs. (FFP Format)	98.85 Secs. (IEEE Format) 17.60 Secs. (FFP Format)
Savage (IEEE)	47.67 Secs./.000000318 Accuracy	119.6 Secs./.000109 Accuracy

There's no contest. Standard benchmark studies show Lattice to be the superior C language development environment. With stats like these, it's no wonder that Commodore-Amiga has selected Lattice C as the official Amiga development language.

Going the distance. You'll experience unsurpassed power and flexibility when you choose from several cost-effective development packages. There is even a full range of supporting products, including a symbolic debugger, resource editor, utilities and specialized libraries.

You'll discover that your software purchase is backed by an excellent warranty and skilled technical support staff. You'll appreciate having access to LBBS—one of the world's first 9600 baud, 24-hour bulletin board services. And you'll be able to conference with other Lattice users through the Byte Information Exchange (BIX) network.

Cross the finish line. Order your copy of the Lattice AmigaDOS C Compiler today. We'll supply the speed. You bring the running shoes.

Lattice
Subsidiary of SAS Institute Inc.

Lattice, Incorporated
2500 S. Highland Avenue
Lombard, IL 60148
Phone: 800/533-3577
In Illinois: 312/916-1600

Digi-Paint

New Tek's 4096 color HAM painting program

by Harv Laser

Digi-Paint has been a long time coming and it has been worth the wait. Advertisements for New Tek's 4096 color HAM painting program first appeared in late 1986, but the program did not appear on dealer's shelves until July 1987. Digi-Paint is capable of working in Amiga's exclusive "Hold and Modify" 4096 color mode.

Digi-Paint comes on a bootable Workbench disk with a well-written and easy to understand 56 page manual that lays flat. My copy also came with a loose piece of paper labelled "Digi-Paint Manual Addendum and Additional Information" which expands on and corrects some of the documentation. Do not lose this sheet. You will miss out on some important features of Digi-Paint if you do. Be sure to fill out and mail your warranty card too, as it will entitle you to a subscription to "The New Tek Times", a newsletter for registered owners of New Tek's products.

Kudos to New Tek for Digi-Paint's packaging. The slick videotape-sized plastic box is durable, easy to store and find on your shelf and protects its contents well. A minor touch but a thoughtful one.

Politely protected

Digi-Paint is copy protected – not the disk, just the program. The disk can be backed up. Each time Digi-Paint loads, you are asked to type in a word from the manual before the program will run. This is probably the least obnoxious form of copy protection. It is understandable that New Tek wishes to sell their program and reap profit from their efforts. Legitimate buyers of Digi-Paint should not have any quarrels with this protection.

"...exciting and unique features you simply will not find anywhere else."

The only drawback is that you must have your manual handy when you want to use Digi-Paint, but this also encourages you to read the manual and discover the program's capabilities. (At a user group meeting Tim Jenison, president of New Tek, demonstrated Digi-Paint and explained that only the first three letters of the word are needed. This is not mentioned anywhere in the documentation.)

Digi-Paint operates in two modes: Low Resolution HAM (320x200 pixels) or Low Resolution Interlace HAM (320x400 pixels) and all of Digi-Paint's features work in either mode, but the Interlace mode uses quite a bit more memory. If you only have 512K you might even have to close your Workbench screen to use some of the Interlace mode's features. Digi-Paint has a tool for doing just that.

Once running, Digi-Paint will multitask providing you have sufficient memory in your Amiga. Even with 2.5 megabytes of RAM in my computer, I could not get Digi-Paint and Deluxe Paint II to run at the same time. They would both load, but touching Digi-Paint's screen with my mouse pointer would completely lock up my Amiga forcing me to reboot. I was able to run Digi-Paint, Scribble!, and a terminal program simultaneously and flip between the three screens working on projects in each, however Digi-Paint uses gobs of CHIP memory, especially if you've loaded two interlace pictures into it, so multitasking with Digi-Paint on a 512K Amiga might prove to be difficult.

You can start Digi-Paint in either mode, from both the Workbench, by clicking on the appropriate icon, and from CLI, by typing the program's name. To multitask Digi-Paint from the CLI, you simply type "RUN DIGIPAINT" to start the program as a background task, or "RUN DIGIPAINT –400" to start it in the Interlace mode. Either method will ask you for the word from your manual, but once running, you cannot switch modes on the fly; you need to quit the program and re-run it to use the other mode, again typing in a keyword.

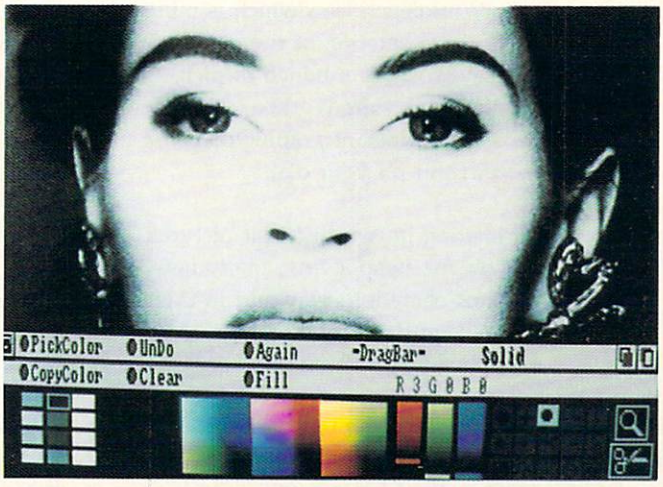
Rather than go into intimate detail about Amiga's different modes of resolution, or bore you with technical jargon as to how the HAM mode works, read the article entitled "Is IFF Really a Standard?" in Amazing Computing Volume 2, Number 7, which has solid information for anyone unfamiliar with the various Amiga graphics modes. Digi-Paint's manual also briefly explains the HAM mode.

Digi-Paint incorporates some advanced dithering techniques that avoid color "shearing" or artifacting.

New Tek is also quick to point out that Digi-Paint is written in 100 percent assembly language for speed.

Tools galore

Once loaded, Digi-Paint presents you with a working environment unlike any other Amiga painting program. The main screen is totally black, and the bottom third of the screen is the menu bar, which you might have expected to be along the top or down the right hand side. The menu is actually another screen that sits on top of the painting screen, like an upside-down window shade, and it can be grabbed with your mouse and dragged up and down, or pushed behind your painting screen to get it out of your way. Bringing the menu back is as simple as clicking your right mouse button at any time.



The menu displays your initial set of drawing tools and the color palette. Many of the tools will appear familiar but some of them will take studying to appreciate. On the left is a 16 color palette. Each of these colors may be clicked on and used to paint with immediately, and a large rectangle to the right of the 16 color palette indicates which color you are currently using. These colors may be changed with a "copy color" gadget to any of the 4096 available colors in the HAM "menu" located to the right. Your mouse pointer also changes to the currently selected color when you begin drawing, and crosshairs appear around the pointer.

The HAM color menu represents portions of the total of 4096 colors you may use. It is three postage-stamp-sized sections representing red, green, and blue, from left to right. By moving your mouse over one of these three sections, you will activate a color mixing function that will display all possible color combinations as related to the currently selected color. Suppose you are working on the red menu; the amount of red will remain constant, while the amounts of blue and green will change depending on where you move your pointer. Likewise, the blue and green menus.

To the right of these menus are the RGB sliders which allow for yet another method of color selection, or fine tuning the color you have chosen. This is all described on the "addendum" sheet. It took me a while, playing with the color menus, to understand this concept. This is one of the beauties of the HAM mode – the ability to work with 4096 separate colors on the screen simultaneously, and to choose any one of those colors. If you've used non-HAM paint programs which limit you to a maximum of 32 colors at once, you'll really appreciate the enormous amount of color flexibility provided in Digi-Paint once you get the hang of using these palettes.

Continuing to the right side of the main menu you'll find the brushes and tools. Some of these will look familiar to you. Different sized and shaped drawing brushes, a line tool, an oval tool (for ovals and circles), and a rectangle tool (for rectangles and squares). These tools can create either outlines or color-filled shapes.

The Magnifying glass gets you into the "fat bits" mode where you can do highly detailed single-pixel work on a portion of your picture. The arrow keys will scroll you around to different areas while your picture is magnified, but there is only one level of magnification and unfortunately, some of the shape tools do not work when in the magnify mode. I found this a drawback when trying to do some very detailed work. I wanted to use the rectangle tool in the magnify mode, but Digi-Paint just wouldn't let me.

Lastly, there is a little pair of scissors. This is easily the best "lasso/brush/shape grabbing" tool I have ever used. Click on scissors and whip it around the screen in any normal or goofy outline you like. Let go of the mouse button and that shape lifts off your picture and becomes a brush. If you do not finish drawing your outline, Digi-Paint will complete it for you with a straight line between the beginning and ending points. Use that

odd shaped custom brush as a rubber stamp, or in combination with any of the special painting effects described below.

Back across the top of the menu are more gadgets. Undo undoes whatever you just did, in any mode. Even if you accidentally erase your entire picture, as long as you have done no other action in the meantime, Undo will bring it back intact. Undo is only one level deep, only undoing your last action.

'Again' will repeat your last drawing action. This is a wonderful tool. If you're playing with shading, or tinting areas of your picture, Again will place the same colors or effects in exactly the same place on the screen as your first action. If you have just halved your screen into a miniature of itself, clicking on the scissors tool and Again will clip out your miniature picture and turn it into a brush. By using Undo and Again in tandem, you can experiment without any danger of permanent damage to your picture until you are happy with the effect. The Again feature is unique to Digi-Paint.

So far, the toolbox of Digi-Paint looks very complete. Whoops, not quite. Something is missing here.... there's no tool for using text or fonts. Sad but true. Digi-Paint has absolutely no method for you to enter text. Theres not even a 'fonts' directory on the Digi-Paint disk. This is probably my biggest disappointment with Digi-Paint. If you want text, you're going to have to use another Amiga paint program that allows text and font input, and save your text as brushes to be imported into Digi-Paint. I hope New Tek includes text in the upgrade to Digi-Paint. Deluxe Paint, Images, even GraphiCraft have it.

More menus, more effects

Activating your right mouse button on the menu bar reveals another set of menus. While holding down your right mouse button, and sweeping from left to right, you will find Digi-Paint's picture loading and saving options, swap screen feature, and printing option. Digi-Paint has a terrific file and directory requestor and it's very fast. Filenames are displayed in one color, and directories in another. To load a picture, you can type in its name, or double-click on its name. If the disk file has an attached "filenote" it will be displayed in a little window. This makes it easy to annotate your pictures to remember what each is without having to load them. When you are working on many different versions of the same picture, this is a timesaver and a nice extra feature.

"But wait! There's more!" When you save your work, Digi-Paint creates a little Workbench icon which is miniature representation of the picture you just saved. Save a bunch of pictures on disk, and your Workbench window becomes an instant graphic database of the pictures on your disk.

By loading in two different pictures, one on the swap screen, many new features come into play and in this mode, Digi-Paint really shines. More on that shortly.

Moving along the menus, brushes may also be loaded or saved. Then we come to the Effects menu. These effects act on the entire active screen, offering you the ability to double the size of your picture either horizontally, vertically, or both (working from the upper left hand corner only), or halving your picture in the same three ways.

You can soften your picture which copies the whole screen and then shifts the entire screen one pixel and averages each colored pixel to the one next to it. (If you colorize black and white Macintosh pictures after converting them to Amiga IFF format with Scott Evernden's great shareware program, MacView, you might find they flicker wildly. The softening feature will remove all the flicker).

Mirror Flip flips the entire screen horizontally or vertically. Switch Half Screen splits the screen in the middle, either vertically or horizontally, and then switches the halves. Selecting the command again reverses this effect.

Next we come to a menu called "Mode" and this is where Digi-Paint "blows the doors off" all other paint programs. These special Modes work in combination with all of the other tools, and also come into play when you have a second picture in your swap screen.

The Modes are Solid, Blend, Tint, Light Tint, Minimum, Maximum, Add, Subtract, Xor, And, Or, and Shading. When you choose a mode, its name pops onto your menu bar so you don't have to guess which mode you're using. The manual says "The best way to grasp how each of these Mode Menu features works is to experiment with them on a picture" and then goes on to give a brief description of each. An entire book could be written just to explain applications for all of these special modes.

My favorite is the Shading mode. It is probably the one that you will use most often. When you select Shading, you see another new set of gadgets on the menu bar, the Dithering gadgets and Shading control. Dithering means mixing of different colors and depending on how you set the gadgets, you can create objects which smoothly blend into each other, or their background. Shading controls where the dithering takes effect.

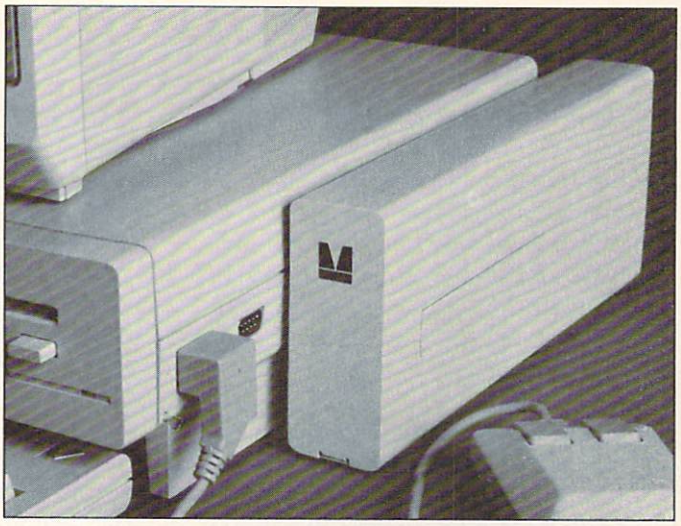
With two pictures loaded, you can create brushes to rub through portions of the top picture to reveal the one below it, dither the two together, crossfade parts of one picture into another... the possibilities are endless. You will play with Shading for hours, discovering new uses for it.

The "Add" mode can be used to tint a black and white picture of a person with realistic fleshtones, and will not

continued on page 12

AVAILABLE NOW!
StarBoard2

Auto-Configuring Fast RAM
Zero Wait States
User Expandable from 512k to 2 Megabytes
Bus Pass-Through
MultiFunction Option: battery/ clock, FPU, parity, Sticky-Disk



If you've owned your Amiga® for a while now, you *know* you definitely need more than 512k of memory. You probably need *at least* double that amount...but you might need as much as an additional two megabytes. We want to urge you to use **StarBoard2** as the solution to your memory expansion problem –and to some of your other Amiga-expansion needs as well!

It's small, but it's BIG–

Since most of you want to expand your Amiga's memory without having to also expand your computer table, we designed **StarBoard2** and its two optional "daughterboards" to fit into a sleek, unobtrusive Amiga-styled case that snugly fastens to your computer with two precision-machined jackscrews.

The sculpted steel case of **StarBoard2** measures only 1.6" wide by 4.3" high by 10.2" long. You can access the inside of the case by removing just two small screws on the bottom and pulling it apart. We make **StarBoard2** easy to get into so that you or your dealer can expand it by installing up to one megabyte of RAM on the standard **StarBoard2** or up to two megabytes by adding in an Upper Deck.

This card has decks!

The basic **StarBoard2** starts out as a one megabyte memory space with 0k, 512k, or one megabyte installed. If you add in an optional **Upper Deck** (which plugs onto the Main Board inside the case) you bring **StarBoard2** up to its full two megabyte potential. You can buy your **StarBoard2** with the Upper Deck (populated or unpopulated) or buy the Upper Deck later as your need for memory grows.

And you can add other functions to **StarBoard2** by plugging in its second optional deck –the Multifunction Module!

StarBoard2: functions five!

If we count Fast Memory as one function, the addition of the **MultiFunction Module** brings the total up to five!

THE CLOCK FUNCTION:

Whenever you boot your Amiga you have to tell it what time it is! Add a MultiFunction Module to your **StarBoard2** and you can hand that tedious task to the battery-backed, real-time clock/calendar. A small piece of MicroBotics software in your WorkBench Startup-Sequence reads the clock and automatically sets the time and date in your Amiga. And the battery *is* included (we designed it to use an inexpensive, standard AAA battery which will last at least two years before needing replacement).

THE FLOATING POINT FUNCTION:

If any one aspect most characterizes the Amiga it's *fast* graphics! Most graphic routines make heavy use of the Amiga Floating Point Library. Replacing this library with the one we give you with your MultiFunction Module and installing a separately purchased Motorola 68881 FPU chip in the socket provided by the Module will speed up these math operations from 5 to 40 times! And if you write your own software, you can directly address this chip for increased speed in integer arithmetic operations in addition to floating point math.

THE PARITY CHECKING FUNCTION:

If you install an additional ninth RAM chip for every eight in your **StarBoard2**, then you can enable *parity checking*. Parity checking will alert you (with a bus-error message) in the event of any data corruption in **StarBoard2**'s memory space. So what good is it to know that your data's messed up if the hardware can't fix it for you? It will warn you against saving that data to disk and possibly destroying your database or your massive spreadsheet. The more memory you have in your system the more likely it is, statistically, that random errors will occur. Parity checking gives you some protection from this threat to your data residing in Fast RAM. Note that the Amiga's "chip" RAM cannot be parity checked.

THE IMMORTAL MEMORY DISK FUNCTION (STICKY-DISK):

When you've got a lot of RAM, you can make nice big RAM-Disks and speed up your Amiga's operations a lot! But there's one bad thing about RAM-Disks: they go away when you re-boot your machine. Sticky-Disk solves that problem for you. It turns all of the memory space inside a single **StarBoard2** into a Memory Disk that will survive a warm-reboot! When your Amiga attempts to grab a **StarBoard2** in Sticky-Disk mode, a hardware signal prevents the system from acquiring the **StarBoard2** as FastRAM (and thereby erasing your files) –instead it is re-recognized as a Memory Disk and its contents are preserved intact. If you want to work rapidly with large files of data that are being constantly updated (such as when developing software) you can appreciate the Sticky-Disk!

Fast RAM –no waiting!

StarBoard2 is a *totally* engineered product. It is a ZERO WAIT-STATE design, auto-configuring under AmigaDOS 1.2 as Fast RAM. Since AmigaDOS 1.1 doesn't support autoconfiguration, we also give you the software to configure memory in 1.1.

Any applications software which "looks" for Fast RAM will "find" **StarBoard2.** And you'll find that your applications run more efficiently due to **StarBoard2** on the bus.

A passing bus? Indeed!

What good is an Expansion Bus if it hits a dead end, as with some memory cards? Not much, we think –that's why we carefully and compatibly passed through the bus so you could attach other devices onto your Amiga (including another **StarBoard2**, of course!).

The sum of the parts...

A really nice feature of the **StarBoard2** system is that you can buy exactly what you need now without closing off your options for future exapansion. You can even buy a 0k **StarBoard2** (with a one megabyte capacity) and populate it with your own RAM (commonly available 256k by 1 by 150ns memory chips). When you add **StarBoard2** to your Amiga you have a powerful hardware combination, superior to any single-user micro on the market. See your Authorized Amiga Dealer today and ask for **StarBoard2**

SUGGESTED RETAIL PRICING:	
StarBoard2, 0k (1 meg space):	$349
StarBoard2, 0k (2 meg space):	$395
StarBoard2, 512k (1 meg space):	$495
StarBoard2, 1 meg (1 meg space)	$595
StarBoard2, 2 megs installed:	$879
StarBoard2, 2 megs & MultiFunction:	$959
Upper Deck, 0k (1 meg space):	$ 99
MultiFunction Module:	$ 99
also available:	
Standard 256k memory card:	$129
MAS-Drive20, 20 meg harddisk:	$1495
MouseTime, mouseport clock:	$ 50

MicroBotics, Inc.
811 Alpha Drive, Suite 335, Richardson, Texas 75081 / (214) 437-5330

continued from page 10

disturb the underlying features. A tutorial in Digi-Paint's manual describes how to do this and a sample black and white portrait is provided for practicing. No other Amiga paint program has anything like these special modes.

The last menu is called Preferences. Clicking on "Close Workbench" permits you to recover the precious CHIP RAM memory your Workbench screen was eating. This is useful if you have limited ram in your system. With 512K, this can make the difference between being able to use the swap screen features of Digi-Paint or not. "No Transparency" makes the black part of your custom brush opaque instead of see-through. "Brush Color Mode" changes a custom brush into your chosen solid color.

With the combination of all of these incredibly powerful features, Digi-Paint becomes your own personal picture manipulating studio. It's uses are limited only by your own imagination, and that is not rhetoric.

I have already mentioned a couple examples, such as converting Macintosh black and white pictures to IFF and then colorizing them. One of Digi-Paint's strong points are its tools provided for colorization, or altering colors on existing pictures.

Digi-Paint is a natural companion to the New Tek Digi-View hardware and software. Digitized pictures are photographic in quality, and Digi-Paint lets you manipulate these photographs in more ways than you can imagine. Instant surrealism is yours. How about creating your own car by cutting, and blending together parts of digitized photos of other cars. Digitize your friends, and combine their pictures with Digi-Paint – switch their heads around, or put your own face on a digitized picture of your favorite movie star's body, or up on Mt. Rushmore. You can really go wild and create all kinds of fantasy pictures, and Digi-Paint's special features make it easy to merge and blend pieces of pictures together seamlessly. You can load in your favorite 32-color IFF pictures and add subtle shading or variations that you simply cannot do with a 32-color paint program. Load in many separate 32-color pictures and they will all happily coexist on the same Digi-Paint screen, even if each picture has its own color palette. By now, you are probably thinking of your own uses, and Digi-Paint makes it all possible.

Digi-Paint does have its limitations. Although it can import a picture of any resolution or containing any color palette, it saves pictures in the IFF HAM mode. You cannot load a Digi-Paint picture into Deluxe Paint, because Deluxe Paint does not understand HAM. If you do, a Digi-Paint picture will be displayed with "garbage" colors. If you want to manipulate a Digi-Paint-created picture with Deluxe Paint, such as adding text, using 3-D perspective effects, and so on, you will need an intermediate processing program to convert the IFF HAM picture into something Deluxe Paint will understand. Programs such as "The Butcher" or New Tek's own "DigiView 2.0" software will do these conversions. Going the other way from Deluxe Paint into Digi-Paint is easy.

Future software

Let's face it – the Amiga can do anything you need a microcomputer to do, and given the right software, can do it more quickly, and elegantly than any other personal computer on the market. But above all, the Amiga is a graphics workhorse, and it is in the field of graphics that this machine really leaves all other PCs behind.

Digi-Paint is a new generation paint program that operates in the 4096 color HAM mode, and is bulging with exciting and unique features you simply will not find anywhere else. At its price, it is a steal. It is not an "everything" program, however, and it does have some faults and limitations.

You cannot directly enter text or use fonts. There is no "Abort" gadget for the printing function. If you want to stop a printout, you have to take your printer off-line and wait for the "printer trouble" requester to pop up and select "Cancel." This is a real nusiance and I hope it is fixed. Although the mouse pointer crosshairs serve their purpose, I would like to be able to turn them off completely and paint with a single pixel brush. I could not find any way to do this.

Occasionally, the part of the menu containing the palettes and gadgets went black – entirely black. I consider this a bug which needs squashing. Clicking a mouse button usually brought the full menu back. It would be nice to be able to switch resolution modes (like Deluxe Paint II) without having to quit the program and reload it. All of the tools should work in the magnify mode, (again, like Deluxe Paint II) not just some of them. If possible, multiple levels of Undo would be nice. Every painting mode generates a "wait" prompt as the screen is refreshed, and your paint is applied. These waits can get quite long sometimes.

Someday, perhaps, someone will create a painting program that does everything. But just as you always add one more number to the biggest number in the universe, you can always think of just one more feature to add to a paint program. I would not hesitate to recommend Digi-Paint as an extremely powerful tool to add to any Amiga artist's arsenal.

•AC•

Digi-Paint $59.95

New Tek Inc.
115 West Crane Street
Topeka, KS 66603

Requires 512K memory.

Sculpt 3D

"Sculpt 3D can render very realistic three-dimensional images on your Amiga, images that **you** design."

by Stephen R. Pietrowicz

People Link CBM*STEVE

Eric Graham, author of the famed Juggler demo, has done it again! This time it's a program called Sculpt 3D. Sculpt 3D can render very realistic three-dimensional images on your Amiga, images that you design. Almost anything you can think of can be modeled; you're limited only by the amount of memory in your Amiga.

The documentation for Sculpt 3D is spiral-bound to lay flat on the desk. The tutorial in the manual gives you step-by-step instructions for each program feature and explains what you should expect from the examples. The tutorial is extremely well done.

Creating objects

The modeling screen for Sculpt 3D has three windows, DOWN, NORTH and WEST. Each window shows how the object you are creating appears if you are looking at it from that particular direction. At first, this viewpoint is a bit confusing, especially if you aren't familiar with 3D rendering packages. With practice, though, it becomes much easier to understand how objects are created and arranged within the scene.

All objects in Sculpt 3D are made up of triangles. This triangular arrangement might sound as though it limits the type of objects you can make, but it really doesn't, as we will see later.

To create an object, you use either a script file, explaining where each vertex and edge is in the picture, or use the mouse to put vertices and edges in each window. How you use the mouse to draw objects is a bit unconventional, but works very well. In normal input mode, not pressing any mouse button keeps the normal mouse sprite on the screen. If the left mouse key is pressed and held down, the mouse sprite becomes a cross hair. By pressing the right mouse key with the left key still held down, you can place a vertex in the scene. By keeping the left key down and pressing the right key repeatedly, you can draw a scene rather easily.

Once vertices are placed in the scene, they are active and shown in yellow. Inactive vertices are shown in purple. Vertices can be connected in several ways. The first method involves placing vertices on the screen three at a time and connecting those vertices by pressing the edge builder gadget in the border of the window. This gadget isn't very easy to use, but it comes in handy for detailed work.

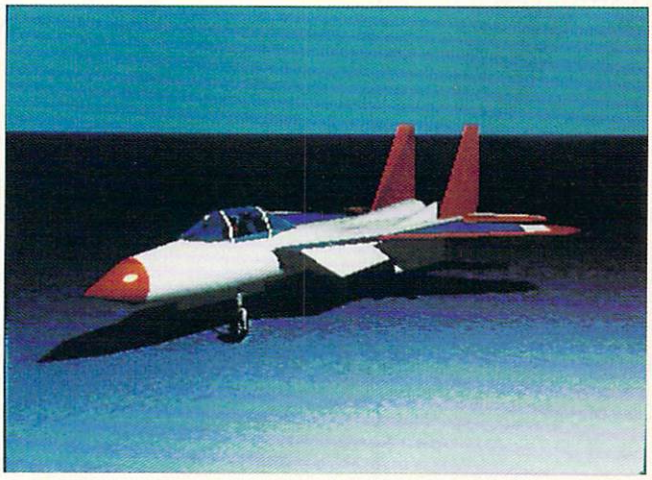
Another method (the method that I prefer to use when making objects), uses one of the drawing tools available under the TOOLS menu. The curve tool lets you draw a series of connected edges within a scene and form those edges into a closed polygon. Once the polygon is closed, you use the "Fill" command to split the polygon into triangles. Fill allows you to draw the shape you want without constructing that object with a series of triangles. Fill takes much hassle out of making objects.

The "magnet" tool works like an actual magnet on the selected vertices in a scene. It can either attract or repel the vertices, depending on the option you select. Vertices closer to the magnet are attracted (or repelled) with greater force. This tool is useful if you have an object you want to make convex or concave, or if you just want to warp the object. I've created labels for cylinders and used the magnet tool to "wrap" those labels around bottles.

The "select" and "deselect" commands allow you to choose series of connected edges by specifying only one point. For example, in a complex scene you might want to extract one object from the middle of the scene and use it. You can select all the vertices in the object by pointing the cursor at each vertex and clicking the left mouse button twice (That's the method you use to toggle a vertex between it's "selected" and "not selected" state).

Using "select," you need only specify one vertex. Sculpt 3D can select the rest of the vertices that are connected with edges to the selected vertex. "Deselect" works in a similar fashion,

but is used to deselect the connected edges of the indicated vertex. "Select" and "deselect" can also be used to activate or deactivate every vertex in the scene. The "select" and "deselect" commands are also available in the TOOLS menu, so you can select or deselect multiple vertices at one time just by moving a small box over the vertices you choose.



You can also grab part of an object and pull it in any direction you wish. For example, you can take the top-most vertex on a sphere and pull it outwards.

By activating all vertices on an object, you can change the location of the object within the scene just by grabbing it and moving it wherever you wish. If you construct a scale model of all your furniture, you can rearrange the rooms and see how things will look!

Numerous other commands let you do just about anything with your objects. Objects parallel to each other can be connected using the "unslice" command. The "reflect" command produces a mirror image of the vertices you have selected. You can spin objects around an axis or create exact duplicates of already modeled objects.

Among the gadgets on the borders of the windows are two gadgets allowing you to rotate objects in three dimensions. Only the active vertices are rotated, so you have control over which objects change.

Some objects are built into the program. They often provide a good place to start making your own objects. The built-in objects are spheres, hemispheres, cubes, prisms, disks, circles, cylinders, tubes and cones. I often take built-in objects to start with an idea for a scene. I then change or build on it. By saving objects that you've modeled for later use, you can build your own library of objects to model more complex scenes.

A color palette that allows colors to be chosen by either RGB (red green, blue) or HSI (hue, saturation, and intensity) is provided so you can choose the colors of your objects. You can also choose the number of bit planes you wish the program to use. Sculpt 3D even has the ability to calculate images with 24 bit planes. While your Amiga can't display images with that many colors (over 16 million) without additional hardware, Sculpt 3D can make those calculations and write them out to a special RGB file.

After you've created your object and chosen its color(s), you can choose a texture. Available textures are DULL, SHINY, MIRROR, LUMINOUS and GLASS. Others, such as a brick, snowy or watery textures would make nice additions to this collection, but you can make quite striking scenes using the provided textures.

When using a mirror texture on a flat surface, I like to draw a solid border around the mirror, making it much easier to see the mirror in the scene. Without a border, it's sometimes hard to tell where the surface of the mirror starts and ends.

You can request that some objects be "smoothed." Selecting the smoothing option blends shades of colors together and softens differences in the surface. Images look less blocky, but they have a tendency to sometimes have a "spray paint" effect. Little dots of color seem to be sprayed across the surface.

The "ground" and "sky" in your scene are under your control as well. You can select the ground to be solid, checkered or not there at all. The sky can be solid, graduated (blended from one color to

continued...

another) or non-existent. You can choose any colors you wish for the ground and sky.

Viewing Objects

To view the scene you've modeled in the three windows, you must choose where you want to view the scene from and what part of the scene you want to focus on. As the manual states, it's almost like pointing a camera and taking a picture. You decide where to center the camera on your picture and where the observer stands to view the scene.

Next, you must choose how you want the lighting (lamps) in the scene to be arranged. Placing lamps in the scene determines how objects stand out and how shadows fall. Bringing the lamp too close to an object washes out the details and the color becomes bright white.

Once Sculpt 3D scenes have been created, they can be viewed with one of four different imaging modes: Painting, Wire Frame, Snapshot or Photo. Painting mode paints the objects (with adjustments for lighting) one color and very quickly reveals how the scene will look very quickly reveals (Normally less than 10–30 seconds, depending on the complexity of the scene). Wire frame mode models the scene with wires and doesn't model any solid surfaces. Snapshot mode performs ray-tracing techniques on the scene. Photo mode also does ray-tracing and shows how the shadows in the scene fall.

Ray-tracing is a method used in graphics to add realism to an object or scene. Rays of light are traced from the observer back to where they originated. If a reflective surface appears in the scene, that is also taken into account. Ray-traced images take longer to compute, but the resulting pictures are quite striking!

Sculpt 3D can compute 5 different sized screens: Tiny, Small, Medium, Full and Jumbo. Tiny images are 1/8 the size of Full, Small images are 1/4, Medium are 1/2 and Jumbo are slightly larger than Full screen images. The smaller the image size, the more quickly it is computed and the less detail is displayed.

Other factors besides ray-tracing and screen size must be taken into account when you try to determine how long an image will take to compute. In particular, glass images sometimes require a very, very long time to compute . . . sometimes several days. A rough estimate of the time an image will take to produce is available and can be shown in the menu bar. It helps to know how much longer you must wait.

The best way to display most of your objects while rendering images is to use the "painting" mode. You'll be able to place your objects in the scene exactly as you want them and get results quickly. Once you've done that, you can move on and experiment with the other imaging modes.

It's best to use smaller screen sizes to get some idea how things look before you try compute the image using a larger screen. Large screens require that more pixels be computed. It's no fun to sit around for a long time just to find out that the image you thought you were going to get didn't turn out the way you wanted. One scene I created with a glass dome on top of a glass cylinder and three floating cubes took two days to render. The program contains those extra screen sizes for a reason. They'll save you time.

ASCII script files can be created for input to Sculpt 3D if you have data that you want to import. Public domain databases are available in ASCII format that you can convert over to the script language. If you had a set of blueprints of a floorplan, you could use that information to make script files.

continued on page 18

NOW SHIPPING

Special

Introductry Price

Complete Communications Package

300/1200 1 Year warranty

300/1200 Fully Hayes compatable Modem - 2 Year warranty

$129.00

(Modem, Cable & Software)

300/1200/2400 Fully Hayes compatible modem CCITT - 2 Year warranty

$249.00

(Modem, Cable & Software)

Call or write for information about our other great products for the Amiga or our Demo disk $5.00

The REASON system is a series of programs designed to aid writers and editors in editing documents.

REASON programs do three things:

*proofread input text *analyze the style of input text *provide help about English usage

Many options give editorial comments and suggestions.

The **REASON** system finds potential errors, then you decide which potential errors need correcting. Thoughtful use of the **REASON** system can help both the experienced and inexperienced writer.

With the REASON **system, there are six main options:**

1. **Prose** describes the writing style of a document, namely, readability and sentence characteristics, and suggests improvements. **Prose** compares a document with standards for one of several document types. INSTRUCTIONAL TEXT will compare input text with good training documents. TECHNICAL MEMORANDA will compare input text with good technical memoranda. And USE CUSTOM STANDARDS will compare input text with any user created standard.
2. **Style** finds sentences that contain passive verbs, expletives, noun nominalizations, and multiple nominalizations. Also, Style will give a readability level for each sentence in the input text or find sentences that are equal to or greater than a specifically defined readability level. Another function performed by style is to find sentences that have a specifically defined length (number of words contained in a sentence).
3. **Word Analysis** will check the input text for general diction, sexist terms, sentences that contain forms of the verb "to be", acronyms and abstract words.
4. **General Structure** checks input text for general organization, general topics, sentence breakdown (parts of speech) and syllable breakdown (syllable count of each word)
5. **Proofread Document** checks for possible spelling errors, double words, possible punctuation errors, diction and split infinitives.
6. **Extra** allows access to AMIGA Preferences and Build Custom Prose Standard.

Requires an AMIGA computer with 512K

AMIGA is a Registered Trademark of Commodore Arniga

$395.00

THE OTHER GUYS

55 North Main Street
Suite 301-D
PO Box H
Logan Utah 84321

(801) 753-7620
(800) 942-9402

VISA MasterCard

continued from page 16

Batch files for computing and saving images are also supported by the program. This feature can be useful if you have a number of images that you want to model. You don't have to be there with the program to save the image and bring in the new image and start it going — Sculpt 3D does that for you.

Overview

Sculpt 3D takes some getting used to, simply because most people aren't used to working in three dimensions when designing on a computer. You don't have to be a mathematical whiz to be able to use Sculpt 3D.

You aren't bombarded with concepts like "3D clipping," what the "painter's algorithm" is or the details of how ray-tracing actually works. In this way, Sculpt 3D doesn't scare you away with it's complexity and is easier to adjust to.

Sculpt 3D is NOT an animation program. Byte by Byte plans to release a program called Animate 3D that will use Sculpt 3D images in animations. Right now, you can only model static frames.

Sculpt 3D multitasks and is not copy-protected. I'm certainly happy that the program doesn't lock up the machine like some programs do. Some ray-tracing scenes I have tried have taken a couple of days to do (Not a very nice thought, especially if you use your Amiga a lot).

Sculpt 3D sometimes gets confused if you pull down a menu while the images in the windows are being redrawn. Parts of the windows don't get refreshed correctly and chunks of the borders are sometimes missing. This problem doesn't occur if you take your time and wait for everything to be redrawn.

If you make a complex scene with many polygons, BE SURE you have enough memory. I have a number of scenes that I can display only in wire frame mode because, otherwise, they Guru the machine. The manual warns against trying to render complex scenes and the program does provide a gauge of the amount of memory left.

Despite some of the problems I've mentioned , Sculpt 3D is one of the best programs I have seen for the Amiga. I've spent hours and hours working with the program and I know I'm going to spend a lot of time with it in the future.

•AC•

Sculpt 3D $99.95

Byte by Byte
9442 Capital of Texas Highway
North Suite 150
Austin, TX 78759
(512) 343–4357

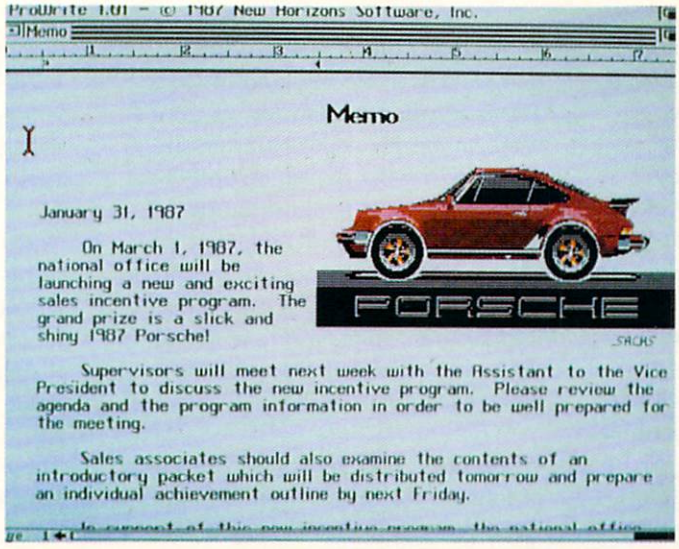
NOT THREE OF A KIND

A brief comparison of Scribble!, ProWrite and WordPerfect.

by Geoff Gamble

What a difference a year makes . . . especially when your interest is word processing software for the Amiga. While writing for AC last year, I lamented the lack of word processing software that would support various fonts for my professional work. Less than a year later, many excellent software packages are available, some of which do allow for font importation (both for screen display and printing).

I had hoped to do a comparison of all available Amiga word/text/publishing packages, but there are just too many to cover effectively in a single review. This review compares many of the features of three such packages: Scribble!, ProWrite and WordPerfect. The review is based upon my own use of each piece of software in my professional writing for at least three weeks. Accordingly, the review tends to focus on the features I use most in my writing.



Sample ProWrite Screen

TEXT MANAGEMENT

All three products handle the basic text features you expect of a good word processor and the basic style features (bold, italic and underlining) are represented on the screen.

ProWrite is a true WYSIWYG processor — what you see on the screen is what will be printed, including superscripts and subscripts. WordPerfect and Scribble! use codes to tell the printer what format is needed, so superscripts and subscripts are not shown on the screen, but they are printed correctly. The format codes in Scribble! appear on the screen, but a special preview mode lets you see what the document will look like when printed. In WordPerfect, format codes are hidden, but can be revealed if necessary.

Both Scribble! and WordPerfect are document oriented, while ProWrite is paragraph oriented. Each paragraph in ProWrite has its own format specification. In Scribble! and WordPerfect, formatting effects all document that follows the codes.

If you wish to change margins and spacing for a particular part of your text in Scribble! or WordPerfect, enter the new margin and spacing codes, type the text and then recenter the original format codes to return to your document format. Table 1 outlines a comparison of many text management features of the three word processors.

Movement through text is similar in Scribble!, ProWrite and WordPerfect. All three use the Amiga's mouse/screen environment, horizontal and vertical scroll bars and arrows. Shortcut codes (key strokes) to move from the top to the bottom of a manuscript, to the beginning and end of lines and most other movements through words, lines, paragraphs and pages are also included.

None of the three packages use particularly fast screen writing routines. If you are working with a particularly large document, the time taken to go from the top to the bottom of the manuscript can be frustrating. Commercial and public domain screen speed up programs can help alleviate this problem.

For writing operations, such as cutting, copying, styling and deleting, you must quickly select a section of text. All three processors allow you to define blocks of text with the mouse, the cursor or with a combination of movement key strokes.

continued...

When using the "Cut" function, the text is held in a buffer and may be "Pasted" back into the document. This feature allows a small measure of security. If you accidentally take out a portion of text, you can recover it by pasting. WordPerfect takes the undelete process even further, by saving up to three deletions in buffers. An "UnDelete" requestor on the Edit Menu allows you to display the last three deleted texts and restore one or more of them in your document.

Much of my professional work involves the use of special characters — namely, a phonetic alphabet. For me a word processor must be able to deal with the many unique fonts, both on screen and in the printer.

ProWrite is ideal, since the standard printing mode is graphics. Although this mode slows down the print speed considerably, whatever you see on the screen is exactly what is printed. With ProWrite, modifying the screen font (I modify the Topazc8), so that the international characters (made by pressing the Alt-C key, plus a letter key) display the phonetic alphabet does the trick. The modified font is saved to the Font: directory of the program disk. ProWrite allows the new font to be called up for use.

Scribble! is more difficult to work with when using exotic fonts. Since Scribble! has many key commands using ALT + key (e.g. cursor movement, toggling Insert mode, certain deletions, etc), the International character set is not available. This arrangement means that I am unable to display both the complete set of regular and a set of phonetic characters on the screen at the same time.

Scribble! does allow for special characters to be sent to the printer. Accordingly, I can access the down loaded printer characters by sandwiching the text to be printed with phonetic characters between codes to start and stop the special fonts. This solution obviously clutters the screen and makes editing more tedious, but it works.

Increased speed is the benefit of using down loaded fonts with Scribble! over the graphics printing of ProWrite. The down loaded fonts are considerably faster and, for large documents like dictionaries, time becomes very important (particularly when the printing operation ties up the document for further editing).

WordPerfect offers several features that work well for my kind of work. The international character set can be used, so the phonetic characters are displayed on the screen. WordPerfect does not directly support the use of new fonts (unlike ProWrite), but you can assign either the special font to be used at startup or use a commercial font program (like FastFonts) to run in the background and change fonts when needed.

WordPerfect also allows down loading of fonts to the printer from within the word processor. In this way, I can down load any needed phonetic fonts at any time. Since codes are hidden from view in WordPerfect, the screen is not cluttered with additional characters. This makes reading and editing the phonetic text quite straightforward.

FILE MANAGEMENT

Management of files is an important part of any word processor. The standard and extra features of the reviewed software packages are summarized in Table 2.

Several extra file management functions (search files, copy files, print files, etc.) are present in WordPerfect, but not in the other two packages. These extras give WordPerfect a "professional" look, but are not crucial in handling text files. I find the rename function particularly useful because I tend to change my mind a lot about what I want to call a particular file.

Although all three packages support ways of changing directories, Scribble! is the easiest. Scribble! allows you to change to other devices and directories, even as one is loading.

With ProWrite and WordPerfect, you must wait until a particular directory is loaded before changing. If you have large directories or are using only one disk drive, you will have to wait it out.

The problem is even more difficult with ProWrite, since you change disks through a Disk gadget in a Documents requester. If you are using several devices (e.g. DF0:, DF1:, HD0:, and VD0:), and wish to change from your

		Table 1													Table 2		
Features	Price	Code/ Mouse	Cut	Paste	Copy	Delete	Erase	Tab-Set /Clear	Insert/ Overstrike	Special Characters	Headers /Footers	Auto Footnotes	Auto Endnotes	Search/ Replace	Load	Save	Delete
Scribble!	$99.95	Both	Y	Y	Y	Y	Y	Y	Both	Y	Y	N	N	Y	Y	Y	Y
ProWrite	$124.95	Both	Y	Y	Y	Y	Y	Y	Insert	Y	Y	N	N	Y	Y	Y	N
WordPerfect	$395.00	Both	Y	Y	Y	Y	Y	Y	Insert	Y	Y	Y	Y	Y	Y	Y	Y

initial device to another, you may be forced to go through a root directory load for each device to reach your choice.

I found WordPerfect's Read File function to be very handy. This functions lets you read the contents of a text file without loading it into the word processor. This feature is pretty nice when you can't remember the contents of a text file.

I put question marks for Scribble! and ProWrite in the File Import row because not all files can be brought into these word processors. If you have a large collection of text files, from many different word processors and perhaps many different machines (that's my situation), it's nice to be able to load files into your current word processor even if the format codes, etc. need to be redone.

PRINT MANAGEMENT

Printing is of central importance in any word processing package and, unfortunately, is a major weakness of many otherwise good pieces of software. The three packages reviewed here handle their printing chores in different ways.

Both Scribble! and ProWrite use printer drivers from the Preferences utility of your Amiga Workbench. You must configure your Preferences file before you use the printing functions of the processor. Since the choices you make concerning the type of printer, paper, etc. will be saved to the disk, you need only go through this procedure once.

Since ProWrite is primarily graphics-oriented, the graphics options of the Preferences Utility is very important and should be used carefully. Make sure you read the printer set-up section of the ProWrite manual carefully before beginning any printing! Printing weaknesses of Scribble! or ProWrite can be attributed primarily to the printer drivers supplied with Amiga workbench. A few of the print features are compared in Table 3.

Unlike Scribble! and ProWrite, WordPerfect uses a set of printer drivers found on its Print disk. In addition, a menu driven printer program allows you to change the configuration of your printer driver in detailed ways. An on-disk Manual explains how the program works, but you also need the printer manual and some patience to make any major changes.

WordPerfect accepts up to six different printer drivers, giving you freedom to change from printer to printer without leaving the word processing program. I have a CItoh Prowriter and a Diablo 630 connected to my A1000 and can switch quickly and easily between the dot-matrix and the daisy wheel.

WordPerfect's true multi-tasking Print function not only shows off the power of the Amiga, but also helps productivity. Printing tasks are handled by a separate program that can run in the background as you continue word processing. This function is handled by print spooling in the PC version of WordPerfect, but with the Amiga version, it is truly multi-tasking.

GRAPHICS

Graphics are the special domain of ProWrite — it's exciting stuff. ProWrite is not only a reasonable word processing program, but it also does a good job of handling graphics produced with any of the Paint/Graphics programs. Bringing a picture into ProWrite from Deluxe Paint is as easy as opening a new window in ProWrite and loading the file from disk. The graphics can then be cut from one window and pasted into another with text.

This past summer, I helped construct a cross-country course for a three day horse event. Using ProWrite and Deluxe Paint II, I was able to write a description of the course with appropriate sketches of the new jumps. Although not mentioned in the ProWrite manual, you should complete all editing on text before adding any graphics when mixing text and graphics. The program does not keep track of the relative position of graphics in relation to text. As you add or delete text material, the text moves, but the graphics do not move with it. Gaps are left where the graphics should be and overlapped text and graphics occur.

DOCUMENTATION

The length of the documentation supplied with each program reflects the complexity of the program. ProWrite is on the easy end of the scale, with a 68 page manual. Scribble! with its WordStar-like dot commands, a speller and mail merge functions, has a 169 page manual.

continued...

Table 3

Rename	List	Change Directory	Read	Import	Drivers	Graphics	Preview	Partial Print	Printer Commands	Multi-Task	Print to File	Print Quality
N	Y	Y	N	?	Prefs.	N	Y	N	Y	N	Y	Codes
N	Y	Y	N	?	Prefs.	Y	NA	Y	N	N	N	Requestor
Y	Y	Y	Y	Y	Special	N	N	Y	Y	Y	Y	Codes

Table 4

Speller	User Dictionary	Thesaurus	Help Files	Widow Tools
40,000+	Y	N	Code Lists	Y
N	N	N	NA	Y
110,000+	Y	Y	Y	N

WordPerfect, with its myriad of special features, a large speller and thesaurus, has a three-ring bound 630 page manual (plus a 30 page printer manual on disk).

In all cases, a tutorial section gets even the novice user acquainted with the program. Good reference sections and indices help you find instructions for the things you keep forgetting. All the documentation is good and very appropriate for each program.

EXTRAS

Given the complexity of each program, it isn't feasible to catalog all of the features that each brings to the Amiga. Several of the extras have been important enough in my own work so that I have listed them in Table 4.

PROBLEMS

Each program has its own quirks, some of which are simply annoying and others which are disruptive to use of the software. I found very few disruptive problems with these three software packages.

Perhaps the most important problem with all three is memory limitation. While all three work on a 512K system, the size of the document is restricted. At least 1 Meg gives better performance and 2 Meg is ideal, particularly if you want the program, supporting tools and the file in memory at the same time.

For instance, the spelling programs for both Scribble! and WordPerfect are disk intensive and take time to check the spelling of even a modest document. Putting the dictionary in memory (or on hard disk) keeps disk access to a minimum and speeds up spell checking considerably. The time taken to check the spelling and make a few changes up to this point in the review was 8.5 minutes with the WordPerfect dictionary on the drive. The time was cut to 2.25 minutes with the WordPerfect dictionary in memory.

A few quirks are worth mentioning. The Speller in WordPerfect uses windows for its word lists and control menu. While the program checks the spelling, a considerable amount of flickering occurs, as the program switches between the document window and the spell Menu window — a bit annoying.

ProWrite also has its quirks. The dead key accents (available with WorkBench 1.2 and listed in the ProWrite manual) are mismapped. ALT-K gives an umlaut, ALT-J produces a tilde and ALT-N is not functional. The Tab, Margin and Indent markers on the Ruler are very difficult to click on when you want to move them. The hot tip of the pointer must be at the left edge and no more than three pixels to the left in order to get these markers moving. I usually end up with a bunch of unwanted tab markers all over the ruler.

At 512K, memory is too limited to send a full screen of data to the printer at one time. Consequently, the printer driver sends a reset command to the printer which can leave thin white lines on the printed page. With some particularly troublesome Preference printer drivers, such as Epson, the white bands seem to alternate with partially printed bands. The driver for Okimate 20 worked flawlessly in this respect.

Scribble! has fewer quirks, maybe because the software has been around longer and all the bugs have been worked out. Printer support is limited to those printers listed in Preferences and suffers whatever problems those drivers have.

The major problem I found with Scribble! is the file size limitation when run on a 512K machine. I usually set the window character limit as high as possible (about 70-73K), and still keep everything working correctly. If you set the size too high, you start losing functions. For example, when I had a document set at 74K, and I had several disk buffers in operation, I was unable to access the archive menu.

CONCLUSIONS

I was very pleased with all three word processing packages. Each effectively handles my text chores, including the special font/character needs of working with non-English languages. Although they are all "word processors" (as opposed to page publishers or text editors), they are very different products.

ProWrite is unique because of its graphics capabilities and its true WYSIWYG capability. If you need to do graphics and text material, this program is ideal. It is easy to learn and fun to use. Its text handling functions are more than adequate for most users, and its color capability is very exciting (particularly if you have a color printer).

Scribble! was one of the two original Amiga word processors and really set the standard. In updated version, Scribble! is an excellent word processing package with mail merge and spelling capabilities. The use of "dot" commands (à la WordStar) seems natural and easy to all of us who have worked on other PC's. If you need a good, solid word processor without all the frills, Scribble! would be my choice.

WordPerfect is clearly the high end of the Amiga word processing line. Most every function you might want in a text package can be found here . . . and maybe a few more. The price is high, unless you can get a copy under one of the special programs offered by the WordPerfect Corporation. With WordPerfect, you get value for your dollar. If you need the processing power, you can get it here.

•AC•

LPD Writer

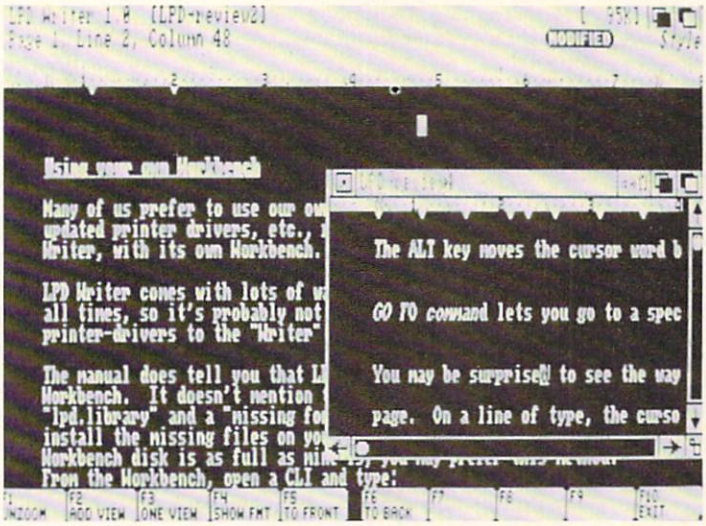
A friendly, easy-to-use word processor

by Marion Deland

I bought my Amiga for the graphics. I already had a "productivity" computer, but I liked using the Amiga and I began to think I'd like a word processor for it — especially if I could find one that's friendly, easy to use, full-featured and reasonably priced.

If you have thought along these same lines, look no further — LPD Writer is finally on the shelves. It's the first in the long-promised LPD series from Digital Solutions, makers of the very popular Pocket series (formerly Paperback) for the Commodore 64 and 128. LPD Planner and LPD Filer are some distance behind — don't look for them for a while.

LPD Writer is a "normal" word processor, not the bit-mapped, desktop-publishing type. You have only one font on the screen — standard Topaz — with all the usual styles, plus superscript and subscript. LPD Writer supports text only, no graphics.



LPD Writer allows you to open windows on one or more documents at the same time

If you are familiar with Pocket Writer, you may expect LPD Writer to be an Amiga version of the same program. Surprisingly, LPD Writer only slightly resembles Pocket Writer. Like Pocket Writer, it is very friendly, in both use and documentation. Format changes are "attached" to paragraphs — a concept that, once you catch on, is very easy to work with. Style commands (underline, italic and bold) are the same as Pocket Writer. Otherwise, LPD Writer stands on its own.

This word processor is one you can just sit down and use. You don't need word processing experience or even Amiga experience. You don't even really need the manual, though it's a breeze to read. Its chief shortcoming, the lack of an index, is largely made up for by a detailed table of contents and organized structure. The manual was written by Karl Hildon, Editor-in-Chief of Transactor Magazine

The program itself has a mild identity problem. It calls itself a "professional" word processor, suggesting business use. LPD Writer is clearly designed for the home user. Little technical information is provided (more about that later, under "Using your own Workbench") and features like column arithmetic and "soft" hyphens and macros, available from most professional word processors used in business, are lacking here. Of course, those programs also cost much more than LPD Writer. The word "professional" is used here as a marketing gimmick, suggesting a product originally designed for use by experts and now available for the rest of us — unnecessary, considering the overall high quality of the product.

LPD Writer does include a spell checker, the ability to work on several documents at once, mail merge, global files, ASCII files, headers /footers and automatic page numbering — everything most of us ever need.

LPD Writer is WYSIWYG ("what you see is what you get"). Everything, even headers and footers, is on the screen. A "Display Format" mode also reveals the spaces, tabs and returns that LPD Writer inserts for you. Some experimentation in this mode is an excellent way to get to know the program and how it behaves.

continued...

Starting out

The program loads with a document "Untitled 1" already open and occupying the full screen. A row of function key labels line the bottom of the screen (You can change all these options by altering the configuration file).

Indicators in the status window fill you in on the name of the current document, position of the cursor, font style, remaining memory and whether the document has been modified since being loaded. A needed piece of info not included is the number of documents you have open.

The function key graphics at the bottom of the screen look a lot like "click" buttons, like the "VCR controls" on the Superbase screen. In fact, I had to overcome a tendency to click in them to perform operations. Nothing terrible happens, but you will have clicked out of your current document, so you must click back in. Until you return, the current document information (file name, page, line and column) will not appear in the status window. Anything you type also will not appear on the screen.

You have three interchangeable ways of operating LPD Writer: the mouse, function keys and a series of control sequences using the CTRL and ESC keys. All three give you access to the same nine menus (PROJECT, MOVE, EDIT, RANGE, FORMAT, SEARCH, STYLE, WINDOW and OTHER) with up to nine options in each.

Each interface has its advantages. Use those you like best.

- The manual recommends the mouse and menus as the best way to learn what LPD Writer can do. They both provide the quickest and easiest way to get results. Switching back and forth to the mouse can be a nuisance, however, when you've got a lot of typing to do.

- I use the function keys a lot. LPD Writer is one of the few commercial Amiga programs that really makes use of the function keys. Even better, you don't have to memorize a set of F-commands for each menu — on-screen graphics give you the current menu's commands.

- The control sequences do have to be memorized to be used effectively, but, in the long run, they are probably fastest and most convenient.

Working with windows

Few commercial programs take advantage of the Amiga's multi-tasking and windowing abilities. LPD Writer fits into this group. You can open as many documents and windows as you have memory for and you can multi-task with ease (A warning is included: Although LPD Writer obeys the rules for multi-tasking, other programs may not, so "Save everything ... or multi-task at your own risk.").

The program defaults to ZOOM mode, causing the document to occupy the entire screen. A small window is the UNZOOM alternative. ADD WINDOW opens another window on the current document. 1 WINDOW ONLY eliminates all but one window for the current document.

I found the multiple documents and windows took some getting used to because few commercial programs make much use of this wonderful Amiga feature. The average user has had much less practice juggling windows than has the "techie." Once you get the hang of it, however, it's great!

A quirk of LPD Writer is that a cursor remains visible in each window, whether current or not — this can be

confusing. The good news is that the cursor marks your place (Just for fun, try opening two windows to the same place in your document. You can type in both windows at once. Cute!).

My habit of using the cursor to tell me which window I'm in got me in trouble. Even now, I occasionally type in a hidden document because I assume the visible window, cursor and all, to be the current document. Worse, I've accidentally closed the wrong file without saving the latest changes (My fault — I wasn't paying close enough attention. If you've modified the file, LPD Writer asks if you want to save it first.)!

SUSPEND, listed under the PROJECT menu, is a related feature. I love this feature! It lets you save everything as is, windows and all, to be RESUMEd later, directly from the Workbench. SUSPEND saves the individual documents, asking you to name those that are untitled. It then saves the positions of windows, etc. in a RESUME file.

When you click on the RESUME file icon, LPD Writer loads itself and the documents and puts everything back in place (except ranges and the contents of the paste buffer) . . . and you're ready to go.

Moving Around

There are several ways to get around in LPD Writer. You can use the mouse or the cursor keys to move around the document. You can move character by character, line by line or window by window. You can also move to the top or bottom of the document. Surprisingly, however, there is no top/bottom of screen command.

The ALT key moves the cursor word by word or sentence by sentence. (ALT/DEL deletes a word—great for a writer!). A GO TO command moves you to any specific page in the document.

You may be surprised to see how the cursor moves up and down the page. On a line of type, the cursor will not go beyond the last typed character. For example, if you position the cursor at the top right of a screen of left-aligned type and then move down, the cursor will zig-zag down the ends of the lines. Also, if you move the cursor beyond the end of your last line, you suddenly find yourself on an empty page — the "end of document." Just move back one space and you'll be back where you came from. Combining the cursor commands with windows to access different parts of the document offers a great deal of flexibility and speed.

Using Tabs

With some word processors (including Digital's own Pocket Writer), the TAB key simply moves your cursor horizontally across the page. With LPD Writer, the TAB key pushes text across to the next tab stop. When you hit the TAB key, an actual character is inserted in the text (you can see it in "Display format" mode). This feature gives you the advantage of being able to re-size columns after you've finished typing. I like it. Both TABs and RETURNs can be erased with a simple BACKSPACE and the text jumps back to follow.

A default series of tabs is set with the "config" file and can be changed to suit your needs. Tabs can also be toggled (individually and together) from within the file. Tabs apply to the entire document and are saved with the document, whether LPD Writer or ASCII format.

Getting It Right

The big advantage of a word processor over a typewriter is that you don't have to get everything right the first time. You can just type in the words

continued...

as they come to you, then go back and polish it up. LPD Writer includes the standard editing and formatting tools:

The RANGE menu allows you to cut, paste and change the font style in blocks of text.

The FORMAT menu commands (margins, justification, spacing, etc.) work paragraph-by-paragraph. Changes apply to the paragraph you're working in and all others that follow, until you make another change. Easy, once you get the hang of it.

The EDIT menu deals with document-wide features, such as spell checking and tab setting/clearing. EDIT also covers headers, footers, page breaks and page numbering. About headers and footers — LPD Writer sets a default header and footer of three lines each. When you open a new header or footer window (even an empty one), the default is replaced.

The SEARCH menu allows you to search for a string of characters from the current cursor position. You can search forward or backward, word by word or sentence by sentence. Extensive pattern-matching allows you to search for and/or replace combinations of words, letters and the "invisible" format characters that LPD Writer inserts.

Checking Spelling

The LPD Writer spell checker consists of a dictionary (a compressed indexed file that cannot be edited) and a User dictionary that you build yourself.

You will need to do a lot of building. The first few times you run the Spell Checker, it queries a word every couple of lines, asking if you want to "skip, add or replace." It saves up the "adds" and enters them in the dictionary when you finish the spell check.

There are some unlikely choices in the LPD Dictionary — and some even more surprising misses. For instance, the word "nothing" is included, but "everything" is left out. "Business," "included" and "available" — common words for a word processor — are also left out.

The inconvenience of having to create part of the dictionary yourself is partly made up for by flexibility. The LPD Dictionary is a file, rather than an entire disk (a larger dictionary would need an entire disk), so, you can have a copy on each data disk, together with a specialized dictionary. No need to swap disks.

Saving and Printing Your Document

LPD Writer saves each file with a Workbench icon. This method has two advantages. First, you can load LPD Writer with a particular file open and ready to go, simply by clicking on the file's icon. Second, you can delete files from the Workbench by putting them in the Trashcan and emptying. In fact, LPD Writer assumes you will do so. Workbench functions like DELETE and INITIALIZE are not included and the reference manual tells you how to multi-task with the Workbench.

When you're ready to print your document, you'll find that LPD Writer takes advantage of multi-tasking there, too. The program prints your document in the background, letting you work on something else at the same time. Nice touch! You can also print just a piece of the document. This feature is great if you find a mistake on the last page of a ten-page document. You can correct your error and print only that page again.

Global printing (printing several files in succession) and mail merge are both available. Mail merge involves printing the same document several times with variations, as may be the case with sending the same letter to

several different people. LPD Writer uses merge data files in regular ASCII format, including those output by database programs like Superbase.

Something that seems to be missing in LPD Writer's print function is a file-name/date/time option. If you use the program only for personal letters, etc., this omission probably wouldn't matter. For a small business, however, or anyone who does successive versions of the same document (like a writer), the deficiency is noticeable. You can insert this information manually in the header, of course, changing it for each version . . . but a word processor is supposed to save you the trouble.

Using your own Workbench

LPD Writer has its own Workbench, including an Epson printer driver. Many of us, however, prefer our own Workbench disks, complete with updated printer drivers, etc., rather than being forced to warm boot a program with its own Workbench. LPD Writer comes with lots of warnings to write-protect the disk at all times, so it's probably not a good idea to copy your own printer-drivers to the "Writer" disk.

The manual does tell you that LPD Writer will load from your own Workbench. It doesn't mention the plaintive requests for "lpd.library" and a "missing font." The requestors tell you to install the missing files on your Workbench disk. If your Workbench disk is as full as mine is, you may prefer this method. From the Workbench, open a CLI and type:

```
ASSIGN lpd.library: Writer:libs/lpd.library
ASSIGN fonts: Writer:fonts
```

With LPD Writer in df0:, load the program. It will look for these files its own disk, instead of on the Workbench disk, and all will be well.

There's one more thing you should know about using your own Workbench. LPD Writer provides a configuration file which can be changed to suit your needs. Certain configuration options, however, including interlace, will be overruled by the Preferences on your Workbench. To work around this difficulty, copy the "LPDwriter.config" file to the "s" directory of your Workbench disk and type:

```
ASSIGN LPDwriter.config: (Workbench
        disk name): s/LPDwriter.config
```

Now the program will get its configuration instructions and corresponding Preferences from your Workbench disk.

In Conclusion

LPD Writer is heavily copy-protected, as you might expect from the company that brought us the Pocket series, which has confounded Commodore 64/128 pirates for years. Unfortunately, the disk-based copy-protection makes the program fragile. We are warned to keep the disk write-protected at all times, except when changing Preferences or the configuration file.

Digital Solutions will provide a backup, recopy your disk or even provide an extra manual at a fee of $18.00 U.S., $23.00 Cdn—reasonable considering the already-low price of the program. If the disk fails within 90 days, they will recopy it free.

LPD Writer does have some areas which need improvement. On the whole, however, LPD Writer is, as the package says, "powerful software that's easy to use."

•AC•

LPD Writer
$119.95
Digital Solutions, Inc.
2-30 Wertheim Court.
Richmond Hill, Ontario
Canada L4B 1B9
(416) 731-8790

Don't miss the boat...

with Amiga expansion products that limit expansion

A500 A1000 A2000

The Advantage™

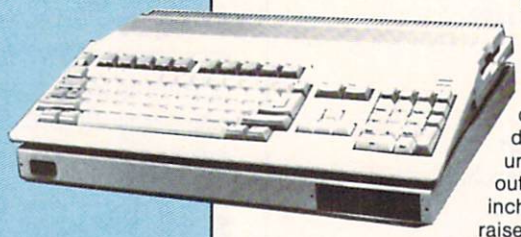
A two megabyte RAM expansion card.

1. No-wait-state design
2. Auto configures with Amiga software
3. Amiga standard design
4. User may install inexpensive 256x1 dynamic RAM
5. RAM disk software survives warm boot. (Not shareware or public domain. Included with purchase of memory board. Available separately for $19.95)

$199

A500

SubSystem™



Use cards designed for the A2000 with your A500, not out-of-date A1000 cards. The SubSystem gives you two expansion slots for A2000 cards and a space for an optional second floppy drive. The SubSystem fits under your 500, completely out of the way. Only 1.5 inches tall, the SubSystem raises the keyboard to the height of an average typewriter and actually makes it easier to use. A UL/CSA approved power supply is included that guarantees additional cards will not overtax the A500. The optional floppy drive is state-of-the-art CMOS design with extremely low power requirements. Cards and disk drive can be easily installed at a later date.

$249

A500 A1000 A2000

DIRECT MEMORY ACCESS **(DMA)** SCSI INTERFACE. Just because you have an Amiga 500 doesn't mean that you don't want the speed of DMA. Using the SubSystem with our A2000 card gives you what others only offer to A2000 owners. No matter what Amiga you own, Pacific Peripherals makes a SCSI for you. Our SCSI offers you compatibility with proven Apple Macintosh external storage devices. As a matter of fact, all of our drives are Macintosh compatible. If you use the SCSI in your A2000 you have an additional bonus... the ability to add a hard drive **inside** your Amiga and still use external devices. In addition to 30 megabyte and 50 megabyte (and larger) drives, Pacific Peripherals offers the Infinity removable media drive. Once you have purchased the Infinity, you have **unlimited capacity.** Each 10 megabytes of memory costs a whopping $18. (Does 100 megabytes for $180 sound more impressive?) With all this capacity you still get 75ms access time.

SCSI card	**$ 249**
SCSI card w/external drives:	
30 Megabyte	**$ 895**
50 Megabyte	**1295**
Infinity	**1295**

A1000

Cage II™

Two-slot expansion box for the Amiga 1000. Uses "zorro" standard cards **AND** passes thru the 86-pin bus for use with nonstandard applications. Power supply included.

$149

(415) 651-1905

Dealer Inquiries Welcome

Policy: Add 3% for VISA or Mastercard. Allow 3 weeks for checks to clear. Send cashiers checks or money orders for faster shipment. California residents add 7% for sales tax. No charge for UPS ground delivery. Next day and 2nd day delivery available. Prices subject to change.

Pacific Peripherals

P.O. Box 14575
Fremont, CA 94539

Desktop VizaWrite

"...not just a word processor, but also a rudimentary desktop publishing system."

by Harv Laser
People/Link: CBM*HARV

Desktop VizaWrite, a new word processing program for the Amiga, offers extreme ease of use in a "What you see is what you get" working environment. VizaWrite offers some powerful new features and, probably, the best Amiga font printout on a dot-matrix printer that I've ever seen.

Desktop VizaWrite (which I'll refer to simply as VizaWrite for the rest of this article) is the most recent release in a long line of software written by British developer Kelvin Lacy. Lacy's previous credits included VizaWrite64, OmniWriter 64, VizaWrite Classic for the Commodore 128 and VizaStar, an integrated spreadsheet and database program for the 64 and 128 computers. VizaWrite is Lacy's first major project for the Amiga and an impressive descendant of his earlier works.

VizaWrite comes in a large plastic case, on one distribution disk with a 130 page spiral-bound manual, sized just right for placement on your desk next to your keyboard. The VizaWrite disk is not copy-protected, so you can easily duplicate it for your personal use, while storing the original disk safely away. VizaWrite should mount easily on a hard drive. Minimum suggested hardware requirements are any model Amiga, with one disk drive and 256K of RAM. The manual cautions, however, that VizaWrite "has a sizeable memory requirement," so extra RAM (above 256K) on your Amiga is recommended.

VizaWrite is not just a word processor, but also a rudimentary desktop publishing system. It can use and print any Amiga font and also allows the importation of graphics. Graphics can then be manipulated to create very impressive printouts.

VizaWrite can be loaded in many ways. Using a standard Workbench icon or typing "VizaWrite" from the CLI does the trick. Since VizaWrite saves each document with its own icon, clicking on a document's icon loads VizaWrite and immediately loads that document, ready for further typing or editing. VizaWrite can also be loaded in "interlace" mode, which gives you twice as many lines of text on the screen at once. All the usual warnings about interlace apply here, though. You must find a good combination of text and background colors, or else you will be forced put up with a wildly flickering display (With typical British reserve, the manual suggests you to select "a gentle choice of colours," when using interlace mode).

When first loaded, VizaWrite looks at the current directory for a file called "VW.configure," for certain default parameters. A sample VW.configure file (which is regular ASCII text and may be edited within VizaWrite itself, or even in AmigaDos' ED text editor) might look like this:

```
Documentsize=40K
Diskdrive=DF1 *
Printerport=PAR
Startfont=(topaz,8)
```

With this file, you can customize VizaWrite to boot up with certain important parameters already set to your liking. VizaWrite automatically looks for VW.configure when loading. You cannot load a different configuration file after the program has already started running, but all the parameters controlled by this file can be set manually from inside VizaWrite. By default, VizaWrite will use your system-configuration file (found in your DEVS directory on your SYS: disk) for its screen colors and mouse pointer design. The pointer can be customized to your personal taste by fiddling with the Workbench Preferences program.

Upon running VizaWrite, you are faced with a gadget laden screen, with an array of pull down menu choices. Vizawrite runs on its own custom screen, rather than using a standard Workbench window. Users familiar with Amiga's multitasking will immediately recognize the familiar "push/pull" gadgets in the upper right hand corner. These gadgets allow you to push VizaWrite's screen to the rear and bring forward screens and windows containing other tasks on which you might be working. The ability to multitask VizaWrite in a useful way depends greatly how much memory you have — the rule of thumb being, "the more the better."

Once you have VizaWrite loaded and running, just start typing. It's really as simple as that. Everything else

continued...

becomes an option to let you customize your document and how it will look when printed.

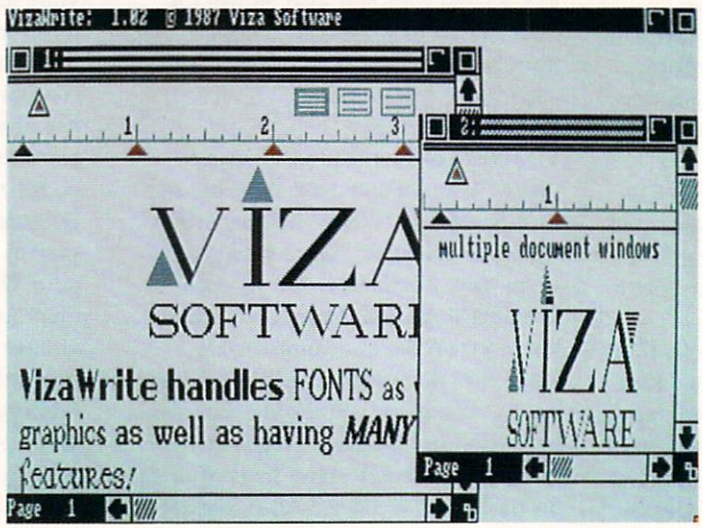
You may notice that VizaWrite uses some unorthodox gadgets on its working screen. The horizontal and vertical scrolling gadgets (which when grabbed with your mouse allow you to move quickly to different places in your document) are diagonally cross-hatched boxes, rather than the usual solid slider "bars." The mouse pointer's "wait" state becomes a red bulls-eye target. The disk and file requesters are also a bit different, but quickly adaptable. They don't really pose any major problems. All the requesters are quite fast and operate logically once you adjust to them.

Requesters that act on displayed text can be moved around on screen, so they don't hide text beneath them. Other requesters (such as the one which displays your font choices) don't need to be scrolled around the screen; they're fixed in place.

VizaWrite wants to load and save files from a directory called "documents." This preference can be a minor nuisance if you don't already have such a directory created. The word "documents" can be changed and any other root or subdirectory name can be substituted, so you are not locked into loading and saving into specially-named directories.

Your first document window opens with a ruler at the top. This ruler (which actually looks like a ruler) controls and displays margins, tab stops, line spacing and line justification (left or right ragged, centered or right justified). The markings on the ruler indicate the actual width of paper (in inches) your printout will occupy. Oddly, the default right margin width is set to five inches, but paper is normally eight inches wide. If left you keep the default setting, the printout will be pushed to the left side of the page. It is easy enough to just grab that margin and move it to a more normal seven inch setting, where the default should really be.

VizaWrite displays text in a slightly magnified mode. Five inches on the ruler is the full width of Vizawrite's default window. This set-up can takes some getting used to. You must remember that standard computer paper is eight inches wide between perforations. As you type, (depending on how fast you type) the speed at which text is written to your screen may be annoyingly slow. This facet of VizaWrite needs improvement, but I'll address upgrades and planned improvements later.



At any point in your text, you can drop in another ruler to alter tab stops, margins and justification. Just grab tabs and margins with the mouse and move them to new positions. Tabs can be removed entirely or more tabs can be added. These new settings will be in effect until you drop in another ruler and reset them.

Rulers can be shown on screen or made invisible. The rulers scroll off the top of the screen as text is entered, so if you forget your current settings, scroll back and check your last ruler. Any ruler can be deleted, except for the initial default ruler. The rulers do not occupy space in your printout and simply provide a visual means of setting parameters.

VizaWrite operates only in the "insert" mode — no "overwrite" mode toggle is available. The delete key deletes the first character to the right of the cursor. The backspace key deletes the first character to the left. The cursor can be dropped into the text at any point and letters, words, lines, parts of lines or whole paragraphs can be cut out, copied or pasted down to other parts of the document using the choices on the top EDIT menu. This menu also controls ruler placement, manually-selected page breaks and the optional header and footer windows. Headers and footers, which appear on each page of a printout, are entered into their own windows and can include page numbering. All the normal menu functions are available when editing headers or footers.

Versatile "search" or "search and replace" features help you locate words or phrases in your text and either jump to them, or replace them with other words or phrases. While writing this article, each time I needed to use the name of the program I simply typed "xxx." When I finished writing, I told VizaWrite to replace every "xxx" with "VizaWrite." This feature saved a lot of typing.

Text styles (bold, underline, italics, superscript and subscript) can be selected individually, or in any combination. Text can be highlighted and its style changed at will. Keyboard shortcuts may be be substituted for the style menu selections and many other frequently used commands in VizaWrite. Speed typists need not constantly reach for the mouse to invoke different text styles.

Shifted tabs can be used to indent paragraphs. Tabs can be controlled by placement on rulers, as explained earlier. Tabs may be aligned by clicking on a desired tab in a ruler, in combination with a keystroke. This feature is useful if you want to enter a column of numbers into your text, aligned on their decimal points, such as:

$125.36 1.95 1995.95 310.79

If you want to enter a column of prices for items in your product line within a letter you are sending to a customer, this method is ideal.

VizaWrite has an extremely versatile font selector. Upon demand, a font requester pops up with all fonts in the assigned font directory sorted alphabetically with the point sizes listed alongside. The fonts' names and point sizes both have their own scrollbars. A newly selected font begins to appear at the current cursor position when new text is entered. Fonts can even be mixed on the same line of text. VizaWrite recalculates tab positions and line lengths, and then realigns the text onscreen to accommodate different sized fonts. I found no limit to the number of fonts allowed in the assigned fonts directory. VizaWrite easily handled extremely long directories of fonts.

VizaWrite can handle fonts up to 255 pixels high, by 255 pixels wide. Vast selections of fonts are available in the public domain and from commercial sources. Varied fonts can add interest and visual appeal to your printouts.

VizaWrite's ability to use and manipulate Amiga fonts is one of its strong points.

Printing Amiga fonts requires a dot-matrix printer (since fonts are bit-mapped you cannot print them on a daisy-wheel printer). While VizaWrite allows you to print in regular draft mode or NLQ (near letter quality) mode (if your printer supports it), its optimized font output is its shining glory. More on that shortly.

VizaWrite can import graphics into your documents. A menu selection governing "images" lets you load previously created graphics from the disk or directory you choose. Since VizaWrite has no facility for creating graphics, they must be created using another program such as Deluxe Paint. VizaWrite works only in black and white, so images should be created and saved using one color ink against one color "paper."

An image brought into VizaWrite is treated like text. It can be cut, copied, pasted elsewhere or removed entirely. Graphics can be sized and shaped by clicking on the image to invoke a frame around it. This frame contains its own sizing, shaping and moving gadgets.

When a graphic image is brought into a document, text is moved to accommodate the image. Text may not be overwritten or "wrapped around" an image. If you want an image to contain text, incorporate the text with the image inside of your paint program before saving.

A few sample images are included on the VizaWrite disk. For the price, it would have been nice to have a disk full of images of clip art or, at least, more than are supplied . . . but making your own images is easy.

Multiple document windows can be opened and pieces of documents can be easily cut and pasted. Whole documents can be appended to others and then re-saved as new documents. The ability to keep many documents open at once, each in its own sizeable and stackable window, is a tied to the amount of memory in your Amiga and the way the VW.configure file is set up.

Saving a document calls up another requester. A warning appears if you try to save a document under the same name as an existing one. A document may be saved in VizaWrite's proprietary format, which will save all special ruler settings, page breaks, fonts, styles, and so on, or may be saved as straight ASCII text. Unlike some other word processors, saving ASCII text from VizaWrite leaves you a very clean file with absolutely NO imbedded commands, page breaks, vast areas of white space, or any other intrusions. Such files will load right into the AmigaDos ED text editor, which most Amigans know balks at loading a file with any binary in it at all. VizaWrite thus becomes a handy tool for preparing text to be ASCII uploaded to an information service or BBS, or for import into any other Amiga software that can read straight ASCII text files.

When a document is saved in document format, it contains information which can be viewed in a special "History" requester. Some of the information is controlled by VizaWrite, other information by the user. The file is time stamped and the last time saved (as set by your Amiga's internal clock) and "version number" (how many times you have saved that document with any changes to it) are displayed. "History" also gives statistics about the document showing a word, sentence and paragraph count and the memory remaining for the document.

Documents may also be stamped with the user's name, as well as being password protected. This may seem like an unneeded feature if you are the only person using your computer. In a multiple user situation or even if

continued...

multiple Amigas are networked together in an office situation (perhaps sharing a common hard drive, file security) VizaWrite is equipped to deal with the situation by allowing documents to be password protected, and by indicating who last saved a document or amended it, and when.

Vizawrite also has mail merge capability. This feature is useful when you wish to send the same document or letter to many different addresses and want each to be an original printout. By creating a data file with names and addresses, and preparing your document to accept these "variables" by marking it with special characters as specified in VizaWrite's manual, invoking the "Print Merge" command will then cause the data file to be read and merged with your main document. Each document to printout until the data file end is reached.

When using any word processor, you may often need to repeat the same words, sequence of words or sections of documents. VizaWrite has a "Glossary" feature which enables the user to build a collection of such oft-repeated text for quick inclusion into any document. An example glossary entry might be your own name, address and phone number, which you certainly type a lot if you print letters on plain paper, rather than letterhead. Multiple glossaries can exist in a "glossary" directory on your disk, enabling you to keep different glossaries at hand for different purposes. You create entries for a glossary by cutting text into VizaWrite's clipboard from a standard document window, and then pasting it into a glossary. This process is neatly handled with requesters and gadgets, as are all of VizaWrite's other powerful functions. Glossary entries can include images and text.

As we've seen so far, VizaWrite has many features. Some of these features overlap with other commercial word processors, while some are unique to VizaWrite. VizaWrite's piece de resistance, however, is the way in which it delivers the final product to a printer. . . which, after all, is what a word processor is for.

VizaWrite differs from most other Amiga word processors in that it does not use the Workbench printer drivers, but instead, has its own built in drivers. These drivers cover a limited range of printers which have been optimized to produce clean, tight, professional looking output of fonts and graphics.

The printers currently supported by VizaWrite are: Commodore MPS1000 and MPS2000, Diablo 630, Epson FX-80 and FX-85, HP Laserjet, and the Juki 6000. Progressive Peripherals says that more drivers will soon be added in an upgrade. It would be highly preferable to own one of these printer models, or one that perfectly emulates those on the list, but good results should be obtainable with many other

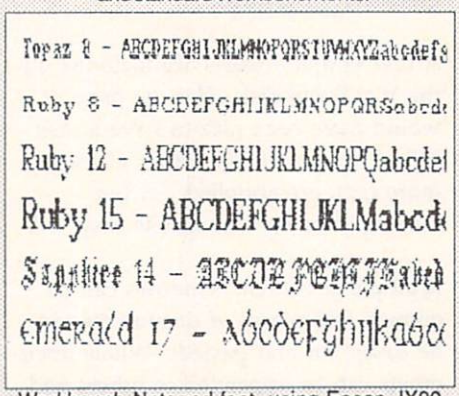
continued on page 34

Printer Drivers:
Workbench vs. VizaWrite

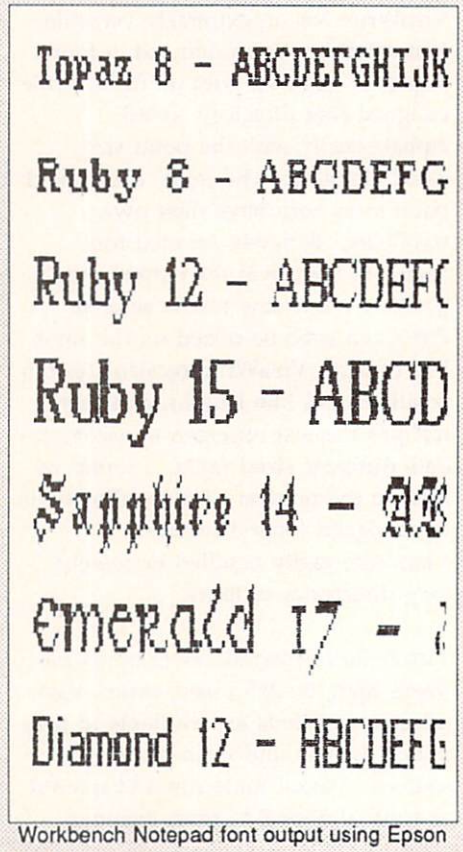
Notice how VizaWrite optimizes Amiga font output, smoothing the jaggies for a very pleasing appearance. By contrast, Notepad's output is jagged and blocky looking, not at all professional in appearance. When sized down to "medium" some fonts printed with Notepad start to fall apart, losing legibility. VizaWrite font output remains extremely legible even on very light fonts such as Opal. VizaWrite does not use the standard Workbench printer drivers, but has its own drivers built in which make this optimization of font output possible.

*Printer used: Epson JX-80 (9 pin dot matrix) and brand new black ribbon.

VizaWrite font using built in Epson FX-80 driver and standard Workbench fonts.



Workbench Notepad font using Epson JX80 driver, sized as "medium" from Notepad.



Workbench Notepad font output using Epson JX80 driver, sized as "auto-size" from Notepad.

Swiss 36

ABCDEFGHIJKLM

abcdefghijklmnop

0123456789! #$%

Symbol 8

ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ
αβχδεφγηιφκλμνοπθρστυϖωξψζ
0123456789!≅#∃%⊥&×()_+|-=\'~¡²¡∀,.<

Times 8

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!@#$%^&*()_+|-=\';':",.<>/?

toto 12

abcdefghijklmnop
abcdefghijklmnopc
0123456789!@ #$%^&

Univers 26

ABCDEFGHIJKLMNOPQ
abcdefghijklmnopqr
0123456789!@#$%^&*

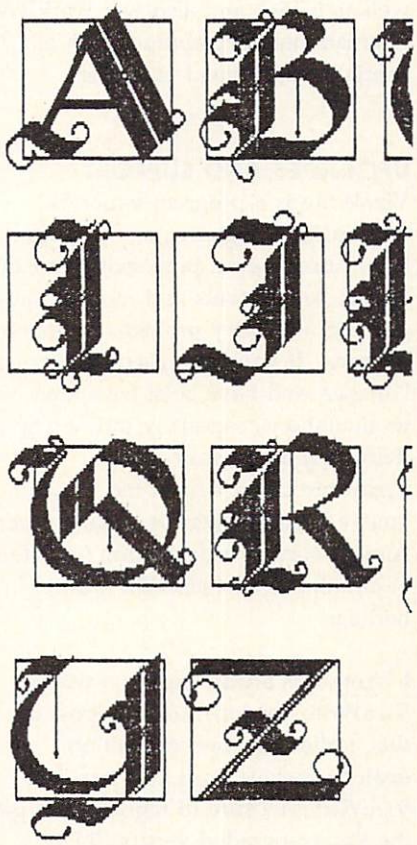
Apple 16

ABCDEFGHIJKLMNOPQR
abcdefghijklmnopqr
0123456789!@#$%↑&*

Alexandra 15

ABCDEFGHIJKLMNC
abcdefghijklmnopq
0123456789!@#$%

Ashwell 34 (below)



Barn 27

ABCDEFGHIJKLMNOPQRSTUVW
abcdefghijklmnopqrstuvwxyz

Cyber 16

ABCDEFGHIJKLMNOPQRS
abcdefghijklmnopqrs
0123456789!@#$%↑&*(

Diamond 12

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!@#$%^&*()_+|-=\'~;'

Emerald 17

abcdefghijklmnopqrs
abcdefghijklmnopqrstu
0123456789!@#$%^&*()_

Flow 8

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!@#$%^&*()_+|-=\' ;':",.<>/

FRANKFURTER 16

ABCDEFGHIJKLMNOPQR
ABCDEFGHIJKLMNOPQR
0123456789!@#$%↑&*

Garnet 9

ABCDEFGHIJKLMNOPQRST
abcdefghijklmnopqrstuvwxyz
0123456789!@#$%^&*()_+|-=\'

Gothic 8

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!@#$%↑&*()_+|-=\';':",.<>/

Granite 15

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!@#$%&*()_+|-=\';':",.<>/?

Sample font output using VizaWrite and a 24 pin printer.

printers. All of my testing was done on my Epson JX-80 printer, which is no longer being manufactured, but was purchased by many early Amiga buyers as a good companion printer. The JX-80 is, basically, an FX-80 with the capability to use four-color ribbons for color text and graphics printout. Although VizaWrite has no provision for color printing, installing a standard Epson type black ribbon in the JX-80, for all intents and purposes, turns it into an FX-80. Thus it functions perfectly with VizaWrite's FX-80 printer driver selection. Sample printouts accompanying this article will give you an indication of just how good VizaWrite's font reproduction is, compared to the "Notepad" program that comes with your Amiga system software.

Fonts are reproduced cleanly, with very little presence of "the jaggies" on curves and diagonals. In contrast, look at the output of the Workbench "Notepad" program, which can also print fonts. Notepad's output looks blocky and crude next to VizaWrite's output. When sized to match VizaWrite's printout, Notepad's font reproduction falls apart. Fonts become illegible and very unprofessional looking.

VizaWrite supports proportional, as well as fixed-width fonts, again to provide the cleanest possible output.

VizaWrite's print requester displays many options for printing documents, including printer driver selection, pitch size, print quality (allowing you to print the bit-mapped Amiga fonts, or use your own printer's near letter quality or draft modes, if you choose not to print using fonts), page range, number of copies, and a choice of continuous feed paper or single sheets. I could find no way, nor did the documentation mention a way, of saving a group of printing parameters to disk so that upon next running VizaWrite, one could avoid having to make all these selections over again.

With the VW.configure file, VizaWrite should be fixed so that printer preferences are loaded automatically each time the program is run.

VizaWrite DOES have a printer-abort function that works immediately. Once aborted, a printout may be re-started or cancelled.

VizaWrite recognizes that Amiga owners exist outside of the USA, so it supports the PAL video standard, as well as NTSC, and also will work with international keymapping, such as special and accented characters.

UPGRADES AND SUPPORT

VizaWrite is a program which is targeted at Amiga owners who want a full-featured word processor that can output Amiga fonts and monochrome graphics in a very professional looking manner. If you have used Workbench Notepad and have been frustrated by its limitations, especially in the way it delivers fonts to your printer, VizaWrite could be just the program you've been looking for. Many other Amiga word processors don't use fonts at all, hinting of their IBM world heritage.

I wrote this article based on using VizaWrite version 1.02. It's possible that earlier versions are still on dealer's shelves, so if you purchase VizaWrite, be sure to register it to get the latest upgraded version. New features have been incorporated into the latest version and you should have the most current for your purchase price. VizaWrite 1.02 can now use user-defined keymapping for function keys.

Workbench icons are not saved with documents if VizaWrite is invoked from the CLI. Printouts incorporating fonts and graphics are now MUCH faster than with the first version (1.00). Quite a few bugs found in the first release have been completely fixed or eliminated. More features and upgrades are planned.

VizaWrite does not have abuilt-in spelling checker. Other spell-checking programs such as "Gold Spell" or "Lex Checker" can be used to spell-check VizaWrite documents if they are saved in ASCII format. A VizaWrite document saved in its proprietary "document" format contains all the formatting information in the first 512 bytes of the file so that the makers of spell checking software should be able to update their products to read VizaWrite's document format directly if they desire. In the meantime, if you own one of these spelling checkers, simply save a document from VizaWrite as ASCII text, spell check it, then re-load it into VizaWrite.

More printers will be supported in future upgrades, the first of which will be the Okidata line of dot matrix printers. A "VizaWrite printer driver builder" is planned for an upgraded release which will allow the user to construct his own VizaWrite drivers. I was also told that the next version of VizaWrite will feature a considerable increase in the speed with which text is printed to the screen as it is entered from the keyboard.

Telephone support-after-sale is available directly from Progressive Peripherals & Software in Denver, CO., and is free to registered owners.

If you want the cleanest, sharpest looking font output currently available with many thoughtful and professional word processing features, and you already own one of the supported printers (or one that closely emulates them), VizaWrite is the program for you.

•AC•

Desktop VizaWrite $149.95

Progressive Peripherals & Software
464 Kalamath Street
Denver, CO. 80204
(303) 825-4144

O to 60 in 3 Seconds

Actually, we're being conservative. The ANIM feature in VideoScape 3-D can play up to sixty frames in *one* second. Real time. Perfect for desktop video production. Perfect for desktop presentation. The ultimate 3-D animation system for the Amiga.

VideoScape 3-D has been designed to work with any Amiga computer using a minimum of 512K RAM. It features solid object generation with hidden surface removal, diffuse reflection from a light source, specular reflection, and a wire frame mode.

VideoScape's Easy Geometry Generator lets you create simple geometric shapes like cubes, spheres, boxes, and cones. You can also use Designer 3-D's visual interface to create unusual shapes. VideoScape 3-D includes a series of objects created by Allen Hastings, as well as IFF foregrounds and backgrounds painted by Jim Sachs and Richard LaBarre. You can generate frames and automatically play them back from scripts, step through each frame one at a time, and use manual or automatic camera motions. VideoScape 3-D will work in multiple resolutions up to 704 x 440 including overscan and interlace.

Combined with other software products, including Aegis VideoTitler, Aegis Animator, Images, Deluxe Paint II, or Aegis Animation Workshop, your animations will shift into high gear. Aegis puts you in the winner's circle! Join the team today!

For more information or your nearest dealer:
(213) 392-9972

To order direct:
1-800-345-9871

AEGIS DEVELOPMENT
2210 Wilshire Blvd., #277
Santa Monica, CA 90403

Aedit

An Absolutely Addictive Amiga Accouterment

by Warren Block

If you've read previous reviews I've written, you may have noticed that I like to put my opinions of a product at the very end of the article. I hope to keep readers in suspense or allow them to reach their own conclusions before they discover my own. Well, that isn't going to be the case with this review, as you may have already noticed by the title. There are so many good things about this program that it would be pointless to try to disguise my approval. Read on anyway and see if you agree.

Why do you need an add-on text editor, anyway? What's wrong with ED, the editor that comes with the system for FREE? A lot. ED ignores the mouse (and Intuition in general), is clumsy and slow, and uses freakish commands that are needlessly complex. If you can't remember which key commands do what, dig out your AmigaDOS manual. There aren't any menus to help. ED is worth just about what you pay for it.

Some Amiga owners still struggle along with ED, though, refusing to buy a replacement editor because ED was FREE. I used to be one of those people, until one day Aedit came into my life and changed my outlook on Amiga text editors forever.

A Few Features

All exaggeration aside, here are a few reasons for my affection for Aedit. I describe just a few because this program completely could, and does, fill up a moderately large book.

Aedit comes in two versions: a 57K "complete" version and a 40K version that has a subset of the full version's features. Both are written completely in 68000 assembler, maximizing speed and efficiency. The smaller version has no menus and instead, uses the function-, Amiga-, and ESC keys for command entry. This version is meant strictly for use with the CLI. Since my Amiga has grown to 2.5 megabytes of memory, I rarely use this version. The following description and comments apply only to the full version.

The editing window is a standard Workbench-type window, complete with gadgets for controlling several options in its borders (Power users take note: the window opens to the full dimensions of the Workbench screen, not just a fixed 640 by 200. With the public domain program Morerows, you can gain extra columns and rows of text; Aedit lets you use this space. The program also supports extra memory. With a megabyte or more, you can edit truly elephantine files).

On the right-hand side of the window is a scroll bar, used for scrolling through the text. Since it is difficult to position the cursor accurately with the scroll bar, it is of little use. Aedit has an unusual feature that regains the wasted space the scroll bar occupies. A command-line option disables the scroll bar, adding another column to the displayed text. It's a shame that more programs don't provide this sort of obvious flexibility.

Entering Text

Aedit defaults to overstrike mode when first executed. Switching to insert mode is as easy as clicking on a gadget or pressing a function key. The Return key merely moves the cursor to the start of the next line down. This can be changed to insert a line (as ED does) with one of the gadgets or function keys.

Some obvious cursor controls are available at this point. The arrow keys, in combination with Alt, Shift, and Ctrl, move the cursor varying distances. As the manual explains, since the Alt key is the lowest on the keyboard, it is the least powerful, moving the cursor only a word at a time horizontally, or five lines vertically. Conversely, the Ctrl key (highest on the keyboard) is the most powerful and moves the cursor to the beginning or end of the line or file (The command structure is loaded with these mnemonic (as in "memory-aiding") devices. For example, the F9 key accomplishes undo changes; the manual says, "A cat has nine lives—the F9 key gives your text nine lives as well." But, I'm getting ahead of myself.).

Cursor movement isn't limited to only these controls, though. By combining Alt, Shift or Ctrl with the Return key, you can move the cursor down a line and to the first non-space character on that line, to the left margin on the current line or up a line. The Tab key is consistent with most Amiga programs, moving forward to the next tab stop. Shift-Tab acts as a back tab,

continued on page 38

moving the cursor to the previous tab stop (as is standard on many other systems). Even the Backspace and Delete keys have new functions added to them. When they are combined with Shift, these keys insert blank spaces at the current cursor position, much like the Insert key on the C-64.

Of course, you can use the mouse to position the cursor or highlight areas of text for block cut-and-paste. After such a mouse operation, the pointer disappears out of your way. For "mouseophobics," this means that a click of the menu button eradicates the pointer; to get it back, just move the mouse.

Editing

Once all your precious text has been entered, lots of options are available for changing it. Block cut and paste is available. You can select the text with the mouse, arrow keys or by line number ranges.

If some text is mistakenly deleted, Aedit's Undo command restores it. Since Undo is incremental, you can Undo changes all the way back to a blank screen. This feature can be a real lifesaver. If you accidentally delete something, just press F9 or select Undo from the menu. You can press F6 to clear the Undo buffer and start fresh from the current text, too.

Editing two files (or more) at one time can be very handy. Just having a reference available to look at while programming can be helpful. Aedit has nine different buffers, with the total amount of text limited only by available memory. When you switch to a new buffer (with a keystroke or menu selection), it is just like running another copy of Aedit—but it doesn't require another 57K of memory, as would a "real" copy of the program. Moving text between buffers is as simple as cutting a section of text, switching buffers, and pasting the text back in.

A "case-folding" option is available to create program source code that is in the correct combination of upper- and lowercase for the specific language you are using. If that language is 68000 assembler, the upper- and lowercase keywords are already built into the program. For other languages, you must create and specify a file that contains the keywords in the correct case. This process can make programming in case-sensitive languages easier. It allows you to standardize capitalization in languages that aren't case-sensitive.

Command Performance

You can give Aedit commands four different ways: keypress, menu selection, gadget selection or an extended command. Extended commands are entered by pressing the ESC key, which displays a command line at the bottom of the screen. You then type your command and press the return key. Most of these commands have names that are easy to remember and make sense — at this point, the full power of Aedit begins to show. Only the first few letters of a command are necessary, helping to cut down on typing. How many letters? Well, it varies from command to command. The quick reference card and the reference manual both show the appropriate abbreviations and you can always type the whole command if neither are handy.

Commands may be repeated over a range of lines or a "conditional repeat" command that is based on search strings can be used (for example, to send every line that contains the word "printf" to the printer).

The command line has other uses, too. AmigaDOS commands can be sent out by prefixing them with the "!" character. If you enter a "?", the last command is redisplayed and can be edited.

Typing a number specifies the line with that number. The cursor will then be positioned at that line. Some simple additions are also made: a period specifies the current line; an addition sign the next line; and (not surprisingly) a subtraction sign signifies the previous line.

The command line is also where you type in search phrases. Prefixed with a slash ("/") or backslash ("\"), these phrases allow you to use Aedit's search and replace functions—and we aren't talking just plain "search for this and replace it with this" here, either. Like other Aedit commands, the very concept of "search and replace" is expanded and reworked. For instance, the slash character works as you might expect a search to perform, but the backslash is equivalent to a "not" search. In other words, "find everything that is not this." A Boolean "OR" operator allows the combination of search patterns and several types of patterns are pre-defined. For example, the "~" character (tilde, under the ESC key) is defined as any non-alphanumeric character. The "^" character matches only at the beginning of a line and $ signifies the end of a line. Searching for "^a$" would find lines that contain only the letter "a", while searching for "dup~" would find "dup=2" but not "duplicate." Additionally, recurring patterns of characters may be specified — the # symbol signifies zero or more occurrences of the following character (identical to its use as an AmigaDOS wildcard character). The "*" means one or more of the following character. Patterns can be combined to great complexity using brackets. All this power may seem too elaborate and unnecessary, but it's a lot like using pattern matching in filenames — once you've used it, it's impossible to live without.

The line number system comes into play here, too. Typing "/honk" on the command line will only find the first occurrence of "honk." How do

continued on page 40

Other Products From The Other Guys	
Reason	$395.00
Omega File	$79.99
Promise	$49.99
KEEP-Trak GL	$49.99
AMT (Amortization Program)	$39.99
Match-It	$39.99
Math-A-Magician	$39.99
Talking Story Book (Christmas Stories)	$39.99
Musicial Slide Show Demo	$ 5.00

Call or write for more information.

SYNTHIA *High Performance Digital Synthesizer*

A state of the art music tool which will:
Create digital **IFF Instruments** for use with nearly all music programs!
Modifying existing **IFF Instruments.** Use *SYNTHIA* on digitized samples to add reverb, wow, and other enhancements.

SOMETHING FOR EVERYONE:

Additive Synthesis - a traditional method which can create almost any type of instrument.
Plucked String Synthesis - simulates plucked strings . . . right down to the 'pluck'.
Interpolative Synthesis - a method which introduces the natural imperfections found in instruments. (Instruments such as brass, woodwinds, pianos, etc.)
Percussion - build your own drum set . . . create any drum you desire.
Subtractive Synthesis - a simple method of creating instruments.
Special Effects - includes filtering, amplification, phasing, waveshaping, amplitude modulation, real reverb, and . . .
IFF Music Player - powerful and compact. Now you can enjoy those songs that needed a memory expansion before! Up to 32 tracks and 32 IFF Instruments! Supports chords, ties, etc.

NOW SHIPPING

IS IT LIVE . . . OR IS IT SYNTHIA?

Synthia uses the latest technology to generate realistic sounding instruments and even the new families of instruments sound real. A real synthesizer on a real computer!
Why buy digitized instruments when you can *SYNTHIAsize* them?

Requires AMIGA 512K

$99.99

THE OTHER GUYS
55 North Main Street
Suite 301-D
PO Box H
Logan Utah 84321

(801) 753-7620
(800) 942-9402

VISA MasterCard

you get to the rest? Remember how the addition and subtraction signs are used? They can be used here, too: "+/honk~" tells Aedit to find the first occurrence of the word "honk," starting with the next line. The minus sign works, too: "-/honk~" will search backwards, starting with the previous line. To make things easier, the unshifted F10 key finds the next occurrence of the search string, in whatever direction you were last using.

Printing It Out

There are few limits on printed output. The current buffer can be printed to the PRT:, PAR:, or SER: device. If that selection isn't good enough, you can enter your own device name, which may include a file name. You can now "print" to a disk file or RAM: or . . . just about anything.

Since Aedit is a general-purpose editor, you can use it to create Startup-Sequence or other script files, or even for word processing. Margins and wordwrap are supported in the word processing mode which is enabled with one of the window's gadgets. Other word processing functions are provided, including reformatting of paragraphs to fit within the current margins. What Aedit doesn't do (gasp!) is format text for output, but this difficulty has been taken care of, though, by two text formatting programs on the disk (actually, different versions of the same program). Text is written with embedded formatting commands and then fed to one of these programs. Any of the public domain text formatters will work, too.

Documentation

A 76-page reference manual, a double-sided 8-1/2 by 11-inch quick-reference card, and a six-page "Very Basic Instructions" flyer are included in the package. Latest updates to the printed matter are included on the disk, along with long and detailed tutorial files.

The reference manual is a "no-frills" affair. It looks like one of those collections of incomprehensible scribblings stolen from the desks of the programmers while they weren't looking—but it isn't. Rather, the manual is a thorough and complete book, including a four-page index and special sections on the "subset" version of Aedit, error messages, and search patterns. Information is presented clearly and thoroughly. Many of the commands are described with the mnemonic techniques mentioned earlier. The manual is not the place to learn the use of Aedit, though—you are best off actually using the program, paging through the tutorial files on the disk and trying the things they suggest.

The quick-reference card is somewhat helpful, although a function key template to fit in the keyboard cutout would be nice. The "Very Basic Instructions" make few assumptions about your level of expertise. Novices would be well-advised to look here first.

In total, Aedit's documentation is complete, readable and thorough.

But That's Not All!

Aedit exists peacefully in either the Workbench or CLI environments. If you start from the Workbench, it will create icons for your files.

In AmigaTrix II, I described a method of modifying binary files, specifically printer drivers, by loading them into AmigaBASIC strings, modifying the strings and then writing them back out as files. With Aedit, this process becomes easy. By using the built-in functions to read and write binary files, you can use all the other features of Aedit, too. These options really convert binary files into text when reading them. The text is then converted back to binary for writing. Binary data is displayed in a dump format, like what you get when using the Type command's "opt h" option.

Like all high-quality programs, Aedit is not copy-protected in any form.

Problems

I do have a few quibbles with the program. After all, nothing is perfect.

First off, there is only one window—and up to nine buffers containing text. The ability to see more than one at a time, in separate windows, would be quite useful.

The current version of Aedit was written to work with Version 1.1 of the Amiga operating system. It works well with 1.2, but leaves out a few features — specifically, auto-enabled string requesters. It's a drag to have to click inside string gadgets before typing text; the 1.2 release allows them to be automatically enabled.

Once in a while, the screen display of one or two lines doesn't match the actual text in a buffer. If you scroll the screen, then come back, the lines of text are shown correctly. This problem may be Kickstart 1.1/1.2 related or it may be an actual bug (there must be one in there somewhere!). In any case, its occurrence is extremely rare.

Conclusion

How can I give my conclusions on this program? I've only scratched the surface of its powerful commands, the consistent method in which they are used, its reliable performance or the sheer amount of thought and effort which its author, Joe Bostic, put into this program? Easy. If you own an Amiga, buy Aedit.

•AC•

Aedit
$40.00 suggested retail

DRM Programs
1329 Arthur Avenue
Las Vegas, NV 89101
(702) 457-9489

WordPerfect
for the Amiga

by Harv Laser
People/Link: CBM*HARV

After much anticipation and a long wait, WordPerfect for the Amiga has arrived. The number one selling word processor for IBM personal computers and compatibles has made the transition to the Amiga with all of its features and functions intact. If you have used WordPerfect on a PC you will feel totally at home with WordPerfect for the Amiga.

WordPerfect is one of the hot topics on the dial-up networks and Amiga Bulletin Boards these days, and with good reason. The release of this program brings, in the eyes of many Amiga-watchers, a new kind of legitimacy to the Amiga.

WordPerfect Corporation has long been one of the premier names in productivity software for the business world, and to see them so wholeheartedly support the Amiga, not with a "junior" version of their word processor but with a product that stands eye-to-eye with the best PC business software leads many Amigans to believe that other prestigious IBM business software publishers will follow suit.

Borland International long ago announced an Amiga version of its much-hearlded language Turbo Pascal, but never released it. Rumors have also spread that Lotus Development, Ashton Tate, and others had Amiga versions of their popular and powerful software packages in the works or ready to go but were taking a wait-and-see attitude concerning the Amiga's sales numbers and its acceptance by the computer-buying public. Perhaps now that WordPerfect Corporation has taken the plunge and spent much time and money developing its Amiga word processor, with more Amiga products to follow, its brethern in the world of business-oriented software will realize that it is time to stop ignoring the Amiga and recognize the machine for its true capabilities. As this is written, the Amiga 2000 has just hit the stores across the country and is selling very well. It should be evident to all major software publishers that the Amiga is here to stay.

WordPerfect for the Amiga is an "industrial strength" software package. Programmed entirely in fast Assembly language, WordPerfect for the Amiga comes with a boxed 3-ring binder containing its four distribution diskettes, over 600 pages of documentation, function key overlays to fit all Amiga models, transparent keytop stickers to help remind the user of special functions similar to the IBM version, and a quick reference card.

The manual contains step-by-step illustrated lessons for the beginner, an installation guide to assist setting up working disks for the first time, a thorough reference section detailing all of the program's hundreds of functions, and complete appendix, glossary, and index sections for further training and reference. WordPerfect can be mounted on and used from a hard drive. Do not let the fact that the program comes on four disks scare you. One disk contains texts for the manual's lessons, another holds printer driver definitions and is rarely needed except during the intial installation. The third and fourth disks contain the main program and the spell checker/thesaurus dictionary, and these last two can even be combined onto one disk, although in doing so you would have to sacrifice some flexibility of using multiple drives.

WordPerfect has been "Amiga-tized." Those used to the PC version can operate it with very little need to ever touch the mouse. Those who prefer the Amiga's Intuition user interface can work with the program using the mouse, pull down menus, and gadgets. The Amiga's ten function keys each carry four different functions, in combination with the CTRL, ALT and SHIFT keys. The HELP key has not been ignored. WordPerfect's help screens are vast and multilayered with virtually every important function explained with just a couple of keystrokes.

Over 200 printers are supported through the use of built in printer drivers. Existing drivers can be modified or new ones created. Multiple documents can be stacked into a job queue and printed in batches while other documents are edited in up to 32 separate windows at once.

Any combination of keystrokes and functions can be recorded as Macros and saved to disk or played back at any time to avoid much repetitive typing. Macros may even be chained together or contain conditional branching.

continued...

Text columns may be defined, similar to a newspaper, up to five across, and displayed on screen and printed. Full indexes and tables of contents for documents are supported, as are footnotes, endnotes, math columns, outlining, and many dozens of other features.

WordPerfect for the Amiga features a spelling checker with a dictionary of 115,000 words including phonetic and word-template look-up, and a Thesaurus for synonym and antonym checking and display. You may create custom dictionaries and even import dictionaries from your IBM since WordPerfect for the Amiga is file compatible with version 4.1 of the IBM program.

With all of these features, what doesn't WordPerfect have? Well it doesn't use Amiga's Fonts, and it has no facility to import and print Amiga graphics. Those capabilities are available in other word processors such as VizaWrite and ProWrite. This is not to say that fonts and graphics are frivilous features. They certainly aren't, but WordPerfect for the Amiga is a very serious piece of software with just about every tool imaginable to suit the needs of those who require a word processor of its power. It can be used on an elementary level as a simple text editor for casual writing chores, or taken to great heights and used to write an entire novel or textbook with all of the flourishes those kinds of jobs require.

If you purchase WordPerfect do not hesitate in sending in your registration cards. The program has already gone through some upgrades since its initial release and is still being tweaked and fine tuned. Registered owners are entitled to be upgraded to the latest release version and should not hesitate to do so.

If you are a student or an educator and can prove it, WordPerfect offers a special pricing plan for you. Consult with your dealer or contact WordPerfect directly for the details.

A full length review of WordPerfect for the Amiga will be appearing soon in these pages.

•AC•

WordPerfect $395.00

Not Copy Protected

Requires AmigaDos 1.2, 512K, and one 3.5" disk drive

Two drives recommended
Extra memory supported

WordPerfect Corporation
288 West Center Street
Orem, Utah 84057
(801) 225-5000

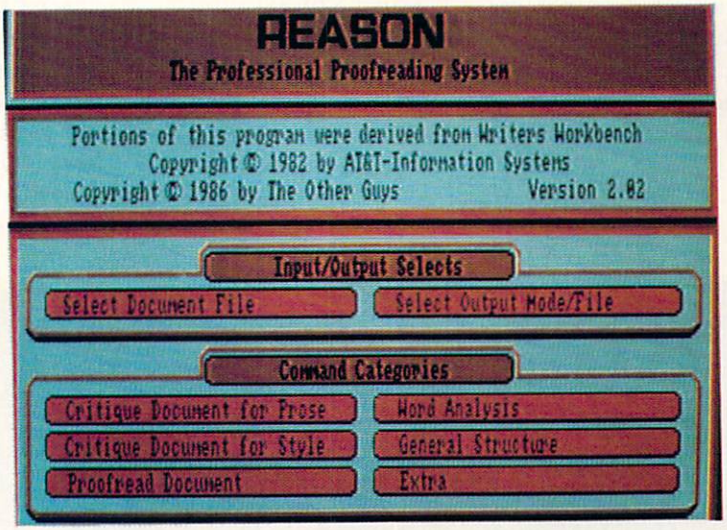
Reason

Analyze your written word

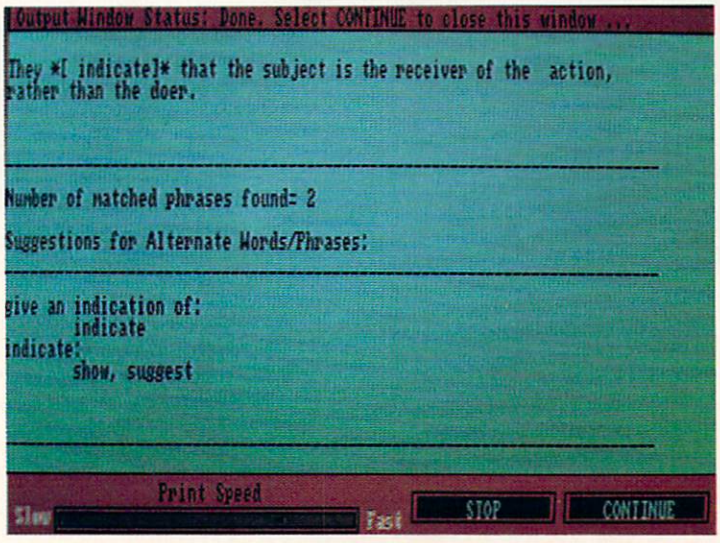
With painstaking care, you wrote your first report. The all night session did not matter. You knew your thoughts, ideas, and conclusions were logical and based squarely on the facts presented. Days later, the paper is returned. Over your neatly typed, expertly arranged thoughts are scrawled red marks and unknown abbreviations. Oh, your thoughts were distinct, but you are criticized for your lack of expertly presented expression.

This scene is played too often. The willingness to express ideas is hampered by the necessity to speak and write clearly.

AT&T faced this same difficulty several years ago. Concerned with the writing ability of their executives, AT&T enlisted the aid of several universities to develop a computerized proof reading program to check text for usage, spelling, diction, and readability. Basing algorithms on popular academic standards, the end result was Writer's Workbench™.



The Reason™ startup screen



The Reason™ output screen

Writer's Workbench™ was so successful, AT&T decided to market the product to Unix and mainframe users in 1984. Universities quickly added the program to their computer systems and incorporated it into their curriculum, all for the reasonable price of approximately $1500.00

Reason

Reason, a $395 two disk software product from The Other Guys, is a fully licensed functional equivalent to AT&T's Writer's Workbench. According to Joseph Nielsen of The Other Guys, this is the first time the program has been available on any micro system. The Amiga™, with its multi-tasking capability, is currently the best choice for Reason.

Reason is a style program made up of 14 other programs running simultaneously in the background. By selecting a text document for proofing, the user has access to over 28 options and the ability to define custom Prose standards and dictionaries.

Continued on page 115

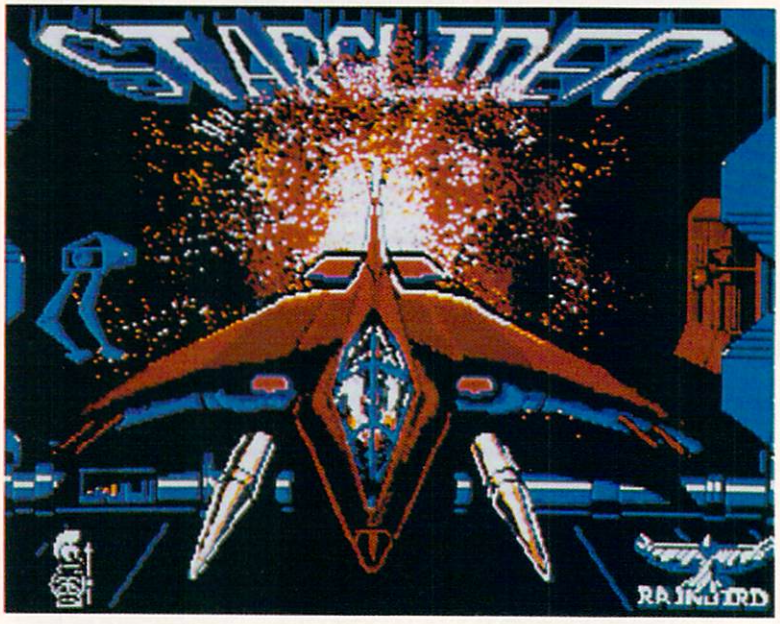
Jez San

The author of StarGlider talks about his background, his work and computing in Europe.

by Ed Bercovitz

It's a "typically British" late winter day in London — cool, damp and slightly foggy. Along famous Oxford Street, the crowds of summer tourists are long gone. At the east end of Oxford, where it crosses Tottenham Court Road, is the centre of London's shopping district for camera and stereo shops, electronics stores and computer dealers. Just a couple of blocks from this intersection is a rather ordinary office building — the corporate headquarters for Rainbird and Firebird Software companies.

Rainbird and Firebird are two of Britain's major software houses. They are both subsidiaries of the government-owned British Telecommunications. Their product line has, until recently, consisted of games for the very large 8-bit computer market in England. In 1986, they brought out their first game for the Amiga, "The Pawn," and followed up this spring with the release of "StarGlider." *Amazing Computing* ™ talked with the author of StarGlider, Jez San, about his background and work, computing in Britain and Europe, and the future of the Amiga in the European market.



AC: So, how did you get into computers and, more specifically, the Amiga?

JS: I've always been interested in them, since day one. I managed to get a cheap Tandy computer when I was 13. I really didn't start programming until I was 15, and even then, it was only little things — patches to other programs and things like that. Then, I bought a BBC computer, which is very popular in Europe, and I did a disassembler and other programs, such as software protection for various companies. Finally, I got a Commodore 64 and started writing various games. I had some unsuccessful products — very good, I might add, but they flopped. So, I helped on a product called Elite, which was Firebird's first major game and that did exceptionally well. It was one of their best sellers of all time. Unfortunately, I was only a consultant on that, so I wasn't in on a piece of the action. They seemed to like my work and they asked me to come up with a game of my own which became StarGlider. [StarGlider] was originally done for Firebird, but when Rainbird split from Firebird, they decided to make Rainbird the "quality" 16 bit-colour software and Firebird the mass market stuff. So, StarGlider got taken into Rainbird. It was their first arcade game and it is still really their only one. They mainly do adventures, like the Pawn and Knight Orc and some which will never get released in America because they are [in] a very "niche" market in England. So, I haven't been in games for very long, but I have been in the business for seven years.

AC: You started working first on StarGlider for the ST, didn't you?

JS: My-16 bit life started when I got a Mac for a couple of months and started programming. As soon as the ST came out, I got one of the first development machines and restarted my work on that, after having got rid of the Mac. I then got the Amiga, but unfortunately the ST work was then so far ahead, that I couldn't suddenly switch to the Amiga. Commodore had promised me a machine a year earlier as part of their original developer program, but they didn't come through.

It was just logistics, [the reason] that StarGlider had to be done on the machine I got first, even though during development the Amiga arrived. But, now I've gotten to do the Amiga version. It was designed with porting in mind. I was conscious of what the Amiga could do, even though I was working on the ST. So, it actually wasn't a very difficult port and the Amiga is actually slightly faster than the ST version, even though the processor is slower.

AC: And now there is a 64 version, isn't there?

JS: Yes — and a Sinclair Spectrum, Amstrad, IBM PC version and various other versions. I did the 68000 ones, but it was my game design for the other versions. The other developers made their own little tweaks for their versions, so people with more than one version might be happy. The Amiga [version] is the last one to come out, unless some other mass market machine comes out.

AC: How do you find working on the Amiga as compared to the ST?

JS: Well, I had a hard system on the ST and I've got a PAL on the Amiga, which is a very nice and fast hard disk. I prefer the Amiga environment because there are more utilities for you to use. And, of course, being able to edit, assemble and so on makes it a very nice environment. It could do with a bit more refinement, but it's still much better than the ST.

AC: Do you work in assembler?

JS: Yes, myself and all my team work in assembly. We don't believe you can write performance software in C. You can for little sprite games — like Marble Madness, where you just have to move the ball around and scroll the screen — but, for 3-D with hidden line movement, it just can't be written in C. Actually, we could, but it would run 4 times slower.

AC: How does this affect your development time?

JS: It does slow down the development a lot because everything has to be absolutely perfect, and when you have a bug, it really shows. Whereas in C, you might be able to get away with a few slight bugs. It also means porting is a lot harder. But, I feel the end result is what counts, and we believe it is worth the time and trouble to program in assembly. This is true of all Rainbird products at the moment, and that's why they have such a good reputation.

One of the unforgivable things I find about the 68000 is that all these 8-bit programmers will write in machine code, and they run very nicely on a 6502 or a Z80. Then, you give them a more powerful processor and, rather than write their stuff in machine code, so it runs faster, they write it in C and let the machine take up the slack. So, the program runs the same speed as it did on an 8-bit machine, where it could have been several times better — a bit faster and a lot smaller. I don't really understand why people write in C.

AC: Do you envision doing things which are more machine specific?

JS: It would be silly to write a game which could only run on one machine. But, having said that, it would also be silly not to take advantage of every facility a machine has. So, as with StarGlider, every individual version is the best that could be done on that machine. On my ST version, everything has to be done in software, like the line drawing, etc. On the Amiga, I, of course, took advantage of the Blitter. So, the Blitter draws my lines and clears areas when I need them, and, in the meantime, the processor can be doing the 3-D rotations.

AC: Do you find it difficult getting programmers for the Amiga?

JS: Yes. 68000 programmers here are very rare, and the ones you find are usually too expensive. Besides which, they know they are good and want to free-lance and don't want to go and work for someone. I try to make my salaries as fair as possible with bonuses and profit sharing, but even then, the company has to survive — which means we have to make more than we pay the programmers.

continued...

AC: You were saying that programmers here are "over the hill" by the time they are 25?

JS: Comparing the same type of computer related jobs in England and America, the people are a lot younger here. In America, it's quite normal to have 30 year old programmers, while here it is quite normal to have 16 or 17 year old programmers and very few 30 year old programmers. I'm not sure why. Possibly it is because our schools are very pro computers and they turn up a lot of "hackers" at an early age. Also, kids really go for computers here and they try and get into more senior positions as soon as they possibly can.

AC: So, most of the programmers at Rainbird are in the 16 to 22 year old age group?

JS: Most of them are certainly under 25. As for the rest of the Rainbird staff, no one (I think) is over 30. The average age can't be more than 22, since there are a lot of 18 and 19 year olds there.

AC: I've heard the comment that computers here are not very well received in business in England — quite the opposite of North America.

JS: Yes, it's much harder here to introduce computers to people who aren't computer literate. We're very polarized here. You're either very pro-computers or very much against them, and there's no in between. The new generation is very much "pro" and hopefully, in a generations time, we won't have this problem. We also have a lot of computers here. We have one of the highest ratios of machines per person of any country. This was because of the "cheap" home computers, like the ZX-81 and products like that. Every school kid has a computer and every school has lots of computers. This is due partially to government funding.

AC: The market here is very different from that in America or Canada. You have machines that we don't see at all in North America, and yet in North America, there's been this great move to IBM PCs and clones for home use.

JS: People here have only just started buying PC compatibles for home use with the Amstrads, because we really didn't have any reliable compatibles here for a long while. And also, no one could afford a real IBM for home. We don't have rich yuppies like in America. So, here a home computer is a 64. Even an Apple is stretching the budget.

AC: What do you see as the future of home computing in the UK.

JS: I'd love to believe Amigas, but, as I said, I don't think they will get their price right. I suspect Amstrad will hold the fort.

AC: With their Amstrad MS-Dos compatibles?

JS: I'm sure that will help them a lot, but they also have their Z80 range which is very cost effective, since it comes with a monitor and disk drive and pretty decent colour and sound. [TheZ80] will still sell for a while to come . . . and no doubt Amstrad [will] have something else up their sleeve for the future — maybe an Amiga look-a-like for half the price.

AC: Commodore has been saying that the Amiga is really doing well in Europe. What's your view on this?

JS: At the the November 1986 developers conference, [Commodore] gave sales figures of 56,000 Amigas in Europe, which I found very hard to believe then. Commodore UK told me at that time that they had sold only 2,500 Amigas, and we must be the second largest market in Europe after Germany. That's because the German price was set and we all had to be in line with them, which, unfortunately, made [the Amiga] too expensive for most other countries. So, assuming, at best, 3,000 have been sold in England, and perhaps 10 to 20,000 in Germany, that doesn't add up to 56,000. Countries like Sweden can't have more than a few hundred. Quite frankly, I was amazed. If they have sold that many, good luck to them . . . but I don't think they have [made the sales].

AC: What is the view of the Amiga here?

JS: There is one problem, especially in England, in that the Amiga is so expensive. Everyone spends their time knocking it, saying it's lovely, but who can afford one? [The Amiga is knocked] because we have a lot of cheap computers here which are still quite good, like the Amstrad. So, until Commodore lowers their price and offers a decent alternative, it's going to get knocked in England. . . as it will in other places.

AC: Is the market here for Amiga games?

JS: No, the game players here can't afford them. They buy Commodore 64s, Amstrads and Spectrums, but the new inexpensive Amiga will help. People are just itching to get Amigas, but they just can't afford them.

AC: So, who is buying Amigas?

JS: Rich hobbyists, programmers (as toys) and people who just have to have state of the art. And, of course, there are the few people who actually need an Amiga — the advertising people for presentations and middle management who have a technical bent, so they can use the machine.

AC: So, as a developer, you must be producing a game which is very much oriented towards the North American market?

JS: Yes, our main sales are coming from America and Germany. But, luckily with an arcade game, there isn't that much country variance. Even in Germany, I believe Rainbird translated the novella into German, but they didn't translate the instructions. The game is still played in English, since there is very little text — it is a pure action game. So, we've quite lucky in that way.

AC: You think an inexpensive Amiga 500 will make a difference in the British market?

JS: Yes, if they price it right . . . but Commodore UK has a habit of pricing things wrong. So, I'd love to believe they are going to get it right this time, but I'm not sure they will. If it came out at 400 pounds, it would be competitive. That would be a nice price and it would sell. Hopefully, it would go down to 300 pounds, if they have competition — but, I'm sure it will be 500 to 600 . . . and people will still need a monitor, since no one will be able to use that type of computer on a TV. There just isn't enough resolution.

AC: It makes you wonder why there is the great difference in prices between the US and the UK.

JS: It's quite simply that for the last couple of years, Commodore USA has always wanted to make a profit from its other foreign subsidiaries. So, at one point, I could buy a machine in the US for a cheaper price than Commodore UK bought their machines for — which is ridiculous. That explains why we have such a high price here. The price here before was 1500 pounds, which was the equivalent of $2300 US. Now, luckily, it's gone down to 1000 pounds, which is still pricey, but it is more acceptable.

AC: As a developer, how do you find working with Commodore UK?

JS: Commodore UK is purely a marketing company. They have only [one] or two technical people left and they are not that technical. So, I find myself phoning America when I have a technical problem and I find that American Commodore and Amiga are very helpful.

AC: Do you perceive any change in Commodore's US attitude towards the Amiga?

JS: I think it will now change because one of the best Commodore people in England has now moved to America. That was Gail Wellington. She's now in the NY office, or perhaps Westchester, coordinating European marketing strategy. She is one very tough cookie and a good person. If she has something to do with the Amiga in Europe, it will do well. Unfortunately, she wasn't given the power she needed last year in Europe.

AC: There was much talk last year that Commodore wasn't "getting their act together. " As a developer, what's your view on this statement?

JS: Well, I still feel that [is the case] quite a lot, but it's nice to know they have the new machines and are thinking on the right lines because I think they have targeted [the new machines] properly.

AC: So, do you think the 2000 will break into the business market here?

JS: I think it has a chance because Amiga software is stabilizing now and we are actually getting decent packages — so I'd like to believe it. I'd certainly buy a few 2000s. I've got a couple of 1000s and I've been holding up buying anymore until the 2000 is out.

continued...

AC: Their strategy seems to be the 500 for the mass market and the 2000 for the business market.

JS: I agree with that and I think that was the right way to go, but I think there might be too much of a price difference between the two. When you think that the 500 to the 2000, with only the addition of a few slots, a better power supply and another 512 k of ram — which they say is only another $150 . . . if it's that easy to upgrade, where does the almost $1000 difference in price come in? It's a lot to pay for just a better looking box.

AC: What do you see as the future of the 1000?

JS: There is no future; it always was an obsolete machine. It was touted wrong, too expensive for the games players and too cheap and [with] not enough features for the business users. So, I think the 2000 and the 500 are ideal machines, so long as they are priced right.

AC: You've become quite well known in the US through your participation on various networks like People Link, Compuserve and Bix. Would you like to comment on that?

JS: Well, I've always been into communications, but it's only recently [that] I've been able to keep in contact with America because it is quite expensive. Our English services are mostly free after hours, so it's very cheap to "online" here. . . but to have to cross the Atlantic via networks and to pay the connect time for the services is quite expensive. I use People Link for downloading software (because it is the cheapest system and it has the fastest downloading with Windowed X-Modem) and, of course, for the fun of it. I get on Compuserve just to keep in contact and I use Bix for information of a technical nature. So, I spread my usage a lot.

AC: I have heard that you are incorporating yourself.

JS: Actually, I already have a company called Argonaut Software. That [company] has been around for 5 years, even before I was serious. StarGlider was done under Argonaut Software. Before it was just my father and myself in Argonaut, but now we actually have a team and an office.

AC: There is a lot of talk about the 68020 option for the 2000 or future 68020-based Amigas. As a developer, would you write programs for such a machine or for the standard Amigas?

JS: In my market, it's different from most, in that being in the games market, you have to aim at the lowest machine. So, we can't take advantage of any facilities which aren't present on the lowest machine. But, with a bit of nice programming, you could check which features are available to the program and accommodate them. With StarGlider, it would probably just make it run faster. There might be times in the future (with my next program) for example, where if it detected that [the game] had a lot more available to it, it might start shading the graphics, rather than just drawing them flat out. But business programmers can [use the extra capability]. They can make their program run only with 9 megabytes of RAM, or some other feature like that.

AC: Do you have another project in the works?

JS: Actually, I have about 3 projects in the works. Depending on how many people I can find for my time, [the projects] will either get done in parallel or in sequence. Rainbird definitely wants a StarGlider II and, although it won't be called that, it will be a follow-up. There will also be some projects that will [be] marketed by Electronic Arts. I'm not sure which one will get done first. As I said, it depends on the people I can get. I've got 2 programmers starting, but I really need about 5.

AC: I've heard that your next projects won't be wire frame, but rather solid graphics programs?

JS: Yes, I have some solid graphics routines that I've been working on, but I haven't really had time to develop them, with finishing StarGlider for various machines. Now, I will finally get a chance to develop them. I am quite sure they will be faster than anything that has been seen so far. They are a lot more mathematically-based, but the end result should be a lot faster. This will probably be seen in StarGlider II and the things I'm doing for Electronic Arts. The games themselves will be totally different. There's no point in doing the same things for 2 companies, but the graphic technique and technology will be [the] same in all of them.

AC: Will you be continuing to work on arcade-style games?

JS: Yes. The three things I currently have under development will be more strategic than StarGlider. StarGlider was a kind of "shoot-em-up" with a bit of strategy. These new things will be a lot more thought-provoking, but still no loss of action. Really, with StarGlider, it more evolved than was designed. It was a case of finding out what the machine could do and keep programming it until it was good. Now, we know what the machines can do, so we will take advantage of it.

AC: Finally, what do you see as your future?

JS: I'm 22 now and I think I will be over the hill in about 2 years time. At the moment, I like the games I'm programming, but I can't see doing this for a lot longer. So, now's the time to get into managing, so that I won't have to be programming to make a living.

AC: Until then, we'll look forward to seeing your programs. Thank you for sharing your views and insights with us.

•AC•

SHADOWGATE

Perhaps the most sensory interactive audio-visual Amiga game yet released.

by Linda Kaplan

You enter the garden. Before you can stop to smell the roses, a ferocious rat races across your feet. The cat is howling not far behind. You examine your environs carefully, experiencing every aspect of the rich graphic display. The flute, apparently unobtainable, is held aloft by the acid, flowing stream of a magnificent fountain. You grasp the flute with just the right preparation and discover it has nested instantaneously in the grip of your magical glove. Joy!

You can play the flute. And you do. Even the least musical of us may carry this treasure around for auditory amusement long after it has served its special purpose. You have rescued the instrument from its elusive perch and have inched your way through the multitude of tests set before you on your mission to save mankind. You have entered the world of Shadowgate, perhaps the most sensory interactive audio-visual Amiga game yet released. Shadowgate, written by ICOM Simulations and published by Mindscape, is a game of awesome special effects.

This honorable successor to the interface offered in Deja Vu and Uninvited is bigger and better than its ancestors. The most important element of Shadowgate is the gratifying interface, which in its action orientation attains many of the qualities of real life. It is possible to complete Shadowgate without having used your keyboard to type even a single four-letter word. Attending to hell-bent adventures such as these does not leave us much time for typing.

OPERATING SHADOWGATE

The Shadowgate interface consists of six open windows which can be moved on the desktop, some of which scroll and grow. A window representing "Thyself" contains your inventory and, presumably, your soul. An "Exits" window depicts all currently available exits. The "inventory" window, which can be enlarged and can scroll, contains the graphic images of the possessions you are carrying.

The text window provides all the supplementary information, such as descriptions of objects to be examined. The largest open window on screen provides a lush, 16 color, visual representation of your current location.

In lieu of typed commands, you have a limited number of verbs at hand (at mouse, actually). Through the Shadowgate menu, you can go, consume, operate, open, close, examine and hit. The most versatile of these activities is "operate."

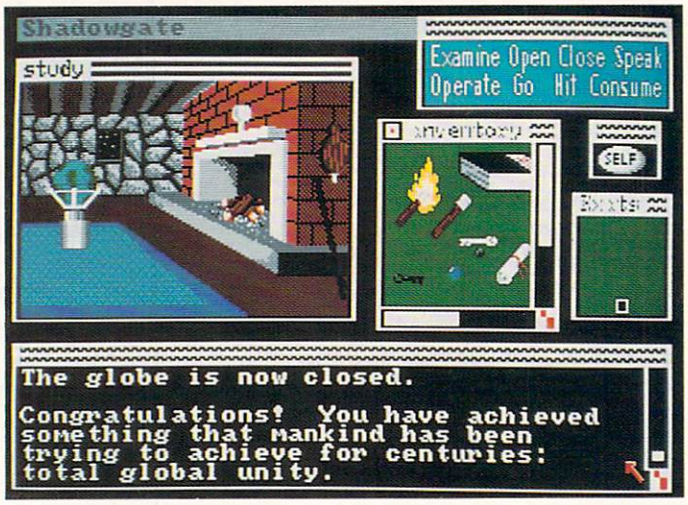
When you don't know what to do, the answer is almost always to "operate." We overlook it because the proper use of this command is initially awkward. However, like learning to use the mouse, it can become preferred.

How does it work? "Operating" causes a wide range of events, and all things "do their thing" when operated. In principle, operating anything will cause it to function in its own unique way. In practice, the functioning of this command isn't always entirely consistent with this theory, but it is close enough to be safely taken as a general Modus Operandi.

You must always operate one thing on another thing. Be sure to keep in mind that you can operate something

continued...

not only on yourself, but also operate the object on itself. After "examining" a book in order to read its title, you can read the contents of a book, either by "operating" the book upon itself or by "operating" yourself upon the book. It is crucial to get the object and subject right. If you operate the book on yourself, rather than vice versa, you won't manage to perform this apparently simple act. In addition, in this particular case, there is a third and simpler way to read a book — just "open" it.



Although this type of shortcut to operating isn't always an option, it usually is worthwhile to experiment with alternatives to operating. Often one can drag one object to another and this action alone will operate it. These moments can be especially delightful. Dragging an object to just the right place can trigger a sequence of thrilling animated events. It may lock satisfyingly into place, initiating eerie sounds which herald the appearance of such things as shimmering crystal balls, suddenly moving walls (revealing mysterious interiors), and hands, which when relieved of gifts, silently sink into the earth.

Some forms of "operate" are obvious. To spear a gnome, for example, just operate the spear on the gnome. To load a sling, you can either open the sling and insert the missile or operate the missile on the sling. Some forms of operating are less obvious. You must examine an open scroll in order to read it. Operating it on something will cause it to perform some magic. To wear a coat, you can operate it on yourself. You may wonder about an inconsistency here. Operating the coat on yourself works. Operating a book on yourself does not (Unless it is a magic book, of course, but that probably isn't what you want to do).

The "open" command in the menu is much less complex, but not always obvious. In addition to the menu choice, when you wish to open either an object or a door, you may simply double-click on that object or door. When objects are opened, their contents always appear in their own window. When doors are open, you can enter them.

As an alternative to using the menu, you can examine objects by simply clicking on them. You can also drag objects to your inventory. What you carry in your inventory and what you can touch with your mouse make up what is available to you at any moment during play.

One element of the interface may be awkward to adapt to. At first glance, the entrances (as they are represented in the Exits window) seem to be placed incorrectly in regard to travelling forward and backward. They aren't improperly placed, but going through the wrong exit can be a deadly mistake. With practice or a bit of thought, you'll develop some comfort with your orientation in this window. However, with the rare exception of hidden exits, you can generally avoid using the Exits window by double-clicking on the place to which you want to proceed. This method of "going" is both more satisfying and less dangerous than using the Exits window.

THE OBJECTS

You can place all moveable objects anywhere on the screen or into your inventory by dragging them with your mouse (A clue: items which highlight when clicked can often be utilized). You can collect more than one item at a time, using a selection rectangle in the inventory and object windows, but not in the display window. Nevertheless, manipulation of the objects on the screen is exciting. Where you place an object is precisely where it will stay . . . unless you

continued. on page 52

NEW for the AMIGA

Precisely™

The Word Processor that doesn't try to be a desktop publisher.

Programs designed for graphic artists are difficult to learn, hard to use, tie up lots of memory, cost too much, and do not handle text very well. Graphics printing on a dot matrix or daisy wheel printer is very slow.

Introducing PRECISELY. The word processor optimized for people who work with words not pictures! Everything you expect in a word processor, such as:

- What you see is what you get (WYSIWYG)
- Very fast screen update/printer speed
- Easy to learn and simple to use
- Converts PaperClip, Pocket Writer and SpeedScript documents
- Online HELP • Mail Merge • Keyboard Macros
- Oops Key to undo mistakes
- Built in spelling checker coming soon

This is PRECISELY the word processor you've been looking for at PRECISELY the right price! Only $79.95 (sometimes less is more). Add $3.00 for shipping and handling, CA residents add 6% sales tax.

CONVERSION SERVICE

Do you have just a few C64/C128 disks of text or data to transfer and no 5.25-inch disk drive on your new Amiga? Use our new Disk Conversion Service to transfer the entire 1541/1571 disk image to a 3.5-inch Amiga disk. Then use Disk-2-Disk, with all of its translation features, to extract and convert individual Commodore files from the 1541/1571 disk image into standard Amiga file format.

The 1541/1571 Disk Conversion Service fee is $6.50 per disk plus a $15* service charge (per order). The fee includes a 3.5-inch diskette, and return shipping via UPS surface.

CAUTION – This conversion service can only process 1541/1571 disks which are formatted in the standard 35-track 256-byte sector format. Disks which are copy-protected, marked with a copyright notice, or formatted with non-standard formats cannot be converted to Amiga format.

* $15 service fee is waived when you purchase Disk-2-Disk with your order.

TRANSFER FILES

TRANSFER C64/C128 files to and from your Amiga!

DISK-2-DISK reads your PaperClip, SpeedScript and Pocket Writer documents or other files on floppy disk directly into your Amiga. Transfers all file types. Use these transferred files with your favorite Amiga programs.

- Reads/writes 1541/4040 and 1570/1571 disk formats.
- Converts Commodore/PET ASCII to Amiga ASCII and vice versa.

TRANSFER MS-DOS and ATARI ST files to and from your Amiga!

DOS-2-DOS reads Lotus 123 worksheets, wordprocessing documents or any other files on floppy disk directly into your Amiga for use with your favorite Amiga programs.

- Reads/writes both 5.25" AND 3.5" MS-DOS disks.
- Reads/writes 3.5" Atari ST diskettes (GEM format).
- Converts ASCII file line ending characters.

Disk-2-Disk requires the Amiga model 1020 5.25" disk drive. Dos-2-Dos runs on any standard Amiga. Disk-2-Disk $49.95, Dos-2-Dos $55.00. Add $3.00 for shipping and handling, CA residents add 6% sales tax.

CCS™

Central Coast Software™

286 Bowie Drive, Los Osos, California 93402 • (805) 528-4906

MasterCard VISA

continued from page 50

operate it in a way that triggers some of the spectacular animated audio-visual special effects of the game.

There is a sense of power about the animated graphic concretization of all we do in Shadowgate; a giant step beyond static illustration which cannot be experienced in a text-only game.

RED HERRINGS

Shadowgate provides scores of elaborate red herrings which introduce a dose of frustration, as well as entertainment. An entire room be a red herring. Crucial items carried faithfully through the game may never be used. One drawback is the fact that the red herrings cannot be identified logically.

"SAVE-AS"ING YOUR LIFE

It can be helpful to "Save-as" the game after making alternate selections, so that you need not return to the beginning each time. Before taking a chance or making a choice of any kind, save-as your game. When you die (as you often will), you can always start from a saved game. It can also be fun to use saved games to review some particularly exciting moments of play.

HINT BOOK

Mindscape provides attractive documentation and, for $5.00 by mail order only, a Shadowgate hint book is also available. The hint book is well-organized and staggeringly comprehensive, offering coded hints to each of hundreds of the problems on three levels. You can select a level of help which preserves much of the game's mystery, a more direct hint, or the actual solution to each problem. The game has its difficult moments and no one should be ashamed of needing some help. I've already responded to more than two hundred requests for help on the telecommunications services.

ENTERTAIN YOURSELF

The final scene includes some of the game's best effects. When the going gets rough and you dread reaching that final scene, you might entertain yourself by playing any of your musical instruments — they all can be played. Try it when it seems there is nothing else to do . . . or, just for the fun of it.

Any game must have something special to make us want to maintain play. In Shadowgate, the staggering quality of special effects and the realism evoked by the graphics and the interface are the keys. The Castle Shadowgate is just as real as any totally fantastic universe can be. In your inventory, heavy objects count more than light ones. Some of your adversaries have a wide variety of

continued on page 54

THE TROUBLE WITH TORCHES

When you begin, you have only a single lit torch with a limited lifetime. When your torch has expired, the screen becomes black, leaving you, literally, in the dark. You then realize that you must find a way to maintain your light source. The odds favor that you will break your neck if you wander around sightless for too long. In addition, you cannot see to play the game. This problem is one of the first you face upon entering the Castle Shadowgate. After you die, having broken your neck in the dark, you begin again.

You eventually notice that before your torch is spent, you receive a warning that it is flickering. You have a specific number of turns in which to locate and light another torch. Remember the "operate" command! If you had a lit torch or any other source of fire in real life and wanted to light another, what would you do? See, that part wasn't all that hard. However, you must be on the alert for traps which are triggered by removing a weight.

CARRYING THE TORCH

Within your environs, you discover fresh torches and even a few eternal light sources. You can carry fresh torches to be used as needed. How many should you carry? There are hundreds of objects in the game and you can tote only a limited number along. An integral element of Shadowgate is the constant conflict of decisions.

You can never be sure if an object once used will be needed again. You'll find a definitive answer to this question in just a few cases, when the objects or their text disappear after use. You carry around many of the over 660 unlikely objects and hope you will need them.

As a result, you cannot bring all the torches you will need to explore the 50 locations in Shadowgate. Only experience will give you knowledge of how to pace torch consumption.

Everything you do (except scrolling our windows) uses a turn, thereby advancing the clock. As a result, exploration of each of the settings consumes much torchlight. Because you cannot see the screen in the dark, position extra torches in easy to find locations in as many rooms as possible (until you are well seasoned). I used the lower right corner of the screen. Should the lights go out, you have a significantly better chance of successfully grabbing a torch and wending your way to one of the sources of eternal fire to light it. No matter how good your visual memory, it is much more difficult to grasp a torch from the middle of the room in the dark.

These Companies Joined Us in New York...

A–Squared
Activision, Inc.
Aegis Development
Amazing Computing
AmiNET
AmigaWorld Magazine
AMuse, Inc.
ASDG, Inc.
Associated Computer Services
Boston Computer Society
Brown–Wagh Publishing
Byte by Byte
Computer Living
Computer System Associates
Crystal Innovations
Designlab, Inc.
Discovery Software
Electronic Arts
Firebird Licensees, Inc.
Fuller Computer Systems
Gold Disk Software, Inc.
Haitex Resources
Hugh's Software Ranch
Impulse, Inc.
Infinity Software, Inc.
Jumpdisk, Inc.
Lattice, Inc.
LICA–Amiga
Magnetic Music, Inc.
Manx Software Systems
MCP Associates, Inc.
Meridian Software, Inc.
Microillusions
Micromagic, Inc.
MicroSearch, Inc.
Microsmiths
Mimetics
Mission Graphics Support
NewTek, Inc.
New Horizons Software
New Wave Software
PComputer Solutions
PiM Publications, Inc.
Psygnosis Limited
R & DL Productions
ReadySoft, Inc.
Sedona Software
Software–of–the–Month Club
Software Insights
Software Visions
Sound Quest, Inc.
Spencer Organization
Spirit Technologies
SubLogic Corporation
Sun Rize Industries
Supra Corporation
Vertex Associates
TeleGames
Very Vivid, Ltd.
WordPerfect Corporation

The Amiga Event!

Join Us in Los Angeles!

The Westin Bonaventure

January 16–18, 1988

For Information on Exhibiting or Attending
Call
Nationwide: 800–32–AMIGA
In New York State: 212–867–4663

continued from page 52

responses. The rooms dim when the torch flickers and brighten when a new one is lit. The screen goes black when our torch has been extinguished or if you close your inventory while carrying the only source of light. (Another hint: If you can still see after your torch is out, there is another light source in the environs.

Light is one of the most exciting elements of the game. There are some awesome pyrotechnics. Once achieved, the deaths of the wraith, the wyvern and the hellhound are sights to behold. Take a look at the dragon's flame in the dark!

In addition to the things that move and the impressive use of light, you may also be entertained by the excellent implementation of sound effects. Some flashes of light are accompanied by thunder claps. Rooms shake convincingly, objects appear, disappear and take wing, and all effects are enhanced by appropriate auditory emphasis. You are pursued throughout the game by a menacing pair of eyes, coupled with merciless mocking laughter. Doors ominously creak, all the instruments can be played, all the weapons can be used and adversaries snarl convincingly. Close to 20 unique sounds are used in 50 different settings.

The game is polished. Wherever the glove lies in your inventory, therein will be placed the flute. Such touches are gratifying.

Humor is an integral element of the general atmosphere of Shadowgate. A favorite comment: "You see the globe open before your very eyes! What an earth-shattering experience!" The game is all in good fun. The humor enriches an already rich experience, consisting of multiple levels of sensory gratification — superior graphics and awe-inspiring special effects.

Shadowgate is not an entirely linear game. There are countless places to explore, objects to discover, problems to solve and many of them are independent of sequence. Of course, once you have mastered the game, you learn that there are more expeditious routes to travel than others. Nevertheless, to some extent, each player experiences his own game.

Unfortunately, Shadowgate is copy-protected. The Amiga was designed for Shadowgate's kind of visual, interactive and auditory fun. Copy-protection is an irritant, even without a hard drive. It is impossible to back up the original disk. If your disk becomes damaged, you must interrupt your devotion to the enterprise to await a replacement. Software publishers underestimate how annoying this situation can be to a player who is enmeshed in the puzzles of a game. To be forced to return to a game, cold, several days later because of the need for a replacement disk can really spoil the game.

Another drawback of this type of game is that it will usually be played only once. You can still enjoy some of the special effects from time to time, after the problems have been solved and the exploration is complete.

A final limitation of Shadowgate is that it is all surface. There is no integrated plot and none of the characters have depth. The task is to "defeat the Warlock Lord and his Behemoth before they darken the world forever." We move from one challenge to another in order to achieve this end. The glitter and display of the surface is more than sufficiently entertaining to compensate for the limitations. In my opinion, the achievement of more continuity and depth, both of plot and of characters, should be the next area for this outstanding team to develop in future games.

•AC•

Bug Bytes The Bugs & Upgrades Column

by John Steiner

Only a couple of weeks have passed since I received my copy of Word Perfect. I have fallen in love with it already. In that short period of time, I have run across several bugs, including problems with the spelling checker and supplemental dictionary. Another bug that bit me several times is in the NewCLI command that is built into Word Perfect. When using a CLI generated from within Word Perfect's NewCLI command, pressing the right mouse button caused an instant lock-up of the Amiga nearly every time.

Authorized Word Perfect dealers have received updated program disks which, according to Lynn LeBaron of Word Perfect's Amiga Development staff, include the following bug fixes. Adding words to the Supplemental Dictionary now works correctly. The Spell program has been fixed. The Copy function in the List files menu works correctly. Miscellaneous bugs in the Print program, especially error handling problems have been corrected.

The NewCLI problem I noticed in the first release version seems to have been corrected, as well. There is a title in the new version CLI window, and it seems to perform flawlessly.

As a software user, I appreciate Word Perfect Corp's upgrade policy. The lack of copy protection makes upgrades easy and convenient for the end user. I trust their faith in the Amiga user community is deserved, and they will not be sorry they did not copy protect their software. To receive an update, visit your Word Perfect Dealer, who can make corrected copies for you, (be sure to bring in your master disks), or contact Word Perfect and report the existence of a bug. Updates to repair bugs are not charged for.

* SubLogic has announced an update to Flight Simulator II, which contains bug fixes, and a few additions. Flight Simulator can now be run from the Workbench, and the copy protection seems to be gone. The single page of information that comes with the new disk notes that now you can run Flight Simulator II from a hard disk or from ram:.

Other enhancements include: Choice of either an analog or digital joystick. The multi-player mode has new features; Setting ADF to '000' makes the needle point in direction of other aircraft, and there is now an autopilot lockon to the other plane. If you replaced your 68000 with a 68010, the program will operate properly. Map zooming allows viewing of a much larger area. The 5 San Francisco ILS approaches now work properly. There is a new procedure for loading the scenery disk. The 'E' and 'W' keys toggle between original scenery and Scenery Disk.

To receive an updated version, return your original disk, and enclose a note requesting the update to the Amiga version of Flight Simulator II to:

SubLogic Corporation
Attn: Amiga FSII Update
713 Edgebrook
Dr Champaign, IL 61820
(217) 359–8482

* Owners of ASDG Incorporated's Facc Floppy Disk Accelerator are invited to send their original disk plus a stamped self addressed envelope in to ASDG for upgrading to FaccII. FaccII is a major upgrade, containing many improvements over the original Facc.

Return your original disk for upgrading since this is your proof of purchase. Enclose a stamped, self-addressed envelope with the disk. First class postage for a disk with a floppy mailer is no more than 56 cents. Canadian owners, can send $2.50 is US funds and ASDG will provide a mailer and postage to Canada. If you live in Europe or Australia, enclose $5.00 in US funds for postage and handling.

ASDG Incorporated
280 River Rd
Suite 54A
Piscataway, N.J. 08854

* Pagesetter, the premier desktop publishing entry for the Amiga from Gold Disk, has an upgrade available. New features include the ability to load, save and delete individual pages, use the entire Amiga extended character set, and allow printout scaling to match printers with a non-standard height/width ratio in graphics modes. The upgrade also contains some minor bug fixes. Cost of the upgrade for version 1.1e is $10.00. The spokesperson for Gold Disk also commented that there has been some confusion about this upgrade due to the imminent release of Pagesetter Professional, a powerful new professional level publishing software. There will be an upgrade path for those who wish to convert from Pagesetter to Pagesetter Professional, however delivery of the new program is not scheduled until fourth quarter of 1987. Once the software begins shipping, you may call Gold Disk and ask about upgrading to the professional version.

Their address is:
Gold Disk, Inc.
c/o Customer Support Box
789 Streetsville
Mississauga, ON Canada L5M 2C2

•AC•

Animation for C Rookies

Part II – Animation Objects

by Michael Swinger

In the first article we discussed a simple method of creating and displaying a bob. Although the program itself was elementary, a simple bob has the potential to create quite sophisticated animation: you can make a bob be any size with as many colors and as much resolution as the screen in which it is displayed; you can move it readily by changing the X and Y coordinates in its associated VSprite structures; you can adjust its display priorities through the Before and After flags in the Bob structures; you can extend its definition through the BUserStuff and BUserExt pointers, and you can create your own cel animation by simply swapping alternate bob images in and out.

Bobs are flexible and powerful, but remember that they are only an intermediate step in the Amiga's animation system. Just as the bob is an extension of a virtual sprite, a bob can be a component of a more complex Animation Component ("AnimComp"), which is in turn part of an Animation Object ("AnimOb"), and by creating a master AnimOb you can have full access to the system animation routines. This linked list of VSprites, Bobs, AnimComps, and AnimObs (which can be daunting at first, but certainly no more so than what you have to go through to create Menus) will allow you to specify the motion and sequence of all of your bobs in advance, and a simple call to Animate will turn everything over to the Amiga's animation system. Of course, you can extend or alter even these elaborate instructions by using the variables in the AnimComp and AnimOb structures to point to your own routines.

I'm afraid we'll have to indulge in a little more RKM bashing at this point. The Rom Kernal Manual does discuss the animation structures at some length, but manages to omit the most crucial information. Certainly a sample program would have helped to explain something as complex as these routines, but there is none in the RKM! (There is an old PD program from Commodore called "Fish", but this muddies the waters even more.) Of all the Amiga literature I have seen, only Eugene Mortimer's *Amiga Programmer's Handbook* mentions the routines, but the author evidently has never worked with them and omits the same information as the RKM. It was not until an article by Roy Thompson appeared in *Ami Project* (Vol 1, #7, April/May 1987) that I discovered why my animation programs didn't seem to work correctly. I encourage you to read this article for a full explanation of the animation structures.

What the RKM doesn't mention, and what Mr. Thompson has unearthed, is that the screen coordinates for the AnimComp and AnimOb structures are specified not as decimal or hex integers, as you might expect, but as 16 bit, fixed point, binary fractions! The RKM does mention this in connection with velocity and acceleration, where a fraction does provide very fine and exact motion control, but to assume that the reader would also know to specify screen coordinates as binary fractions is ludicrous!

In brief, the fraction that you are expected to know about intuitively is a string of 16 0's, with the decimal point fixed at the sixth position from the right: 00 00 00 00 00. 00 00 00 . To indicate a one, it would be written as 1.000000, which is actually decimal 64 to us mortals. Think of the number as being shifted left 6 bits (which is how the machine sees it) and you will see that the actual numbers that you specify are multiples of 64. So, to specify a 1, you enter 64; a 2 is actually 128, etc. It is simpler to enter the numbers as actual screen coordinates and multiply them by 64—130 * 64 would place something at coordinate 130, for example.

The AnimOb structure requires this kind of fraction in the AnY and AnX variables (the actual screen coordinates of the AnimOb which override the coordinates in the VSprite structure), the YVel, XVel, and YAccel XAccel variables (the velocity and acceleration of the AnimOb), and the RingYTrans and RingXTrans variables (which specify the values by which the registration point changes when you set the RINGTRIGGER flag in an AnimComp). The AnimComp structure uses a fraction in the YTrans and XTrans variables—these are the actual pixel offsets from the AnY and AnX coordinates you have specified in the controlling AnimOb. Got it?

The sample program below simply places two bobs (which have become 2 AnimComps of a single AnimOb) on screen and then moves them and

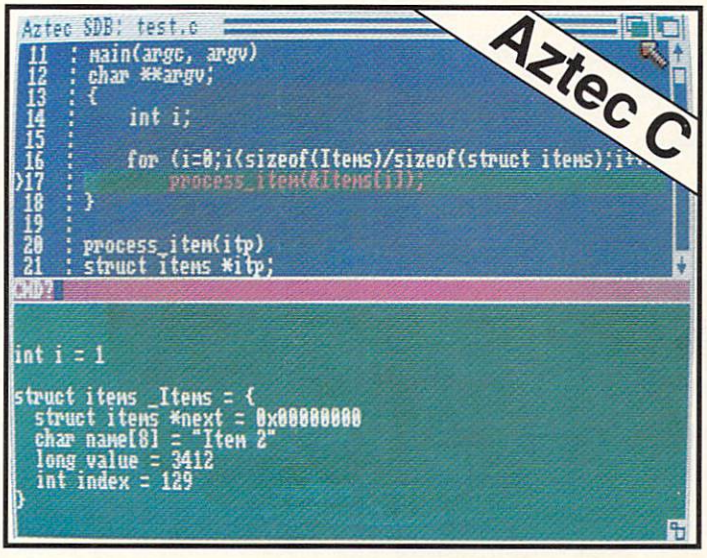
continued on page 58

SOURCE LEVEL DEBUGGER

Announcing the Manx Aztec C Source Level Debugger for the Amiga!

NOW THE MOST ADVANCED COMPUTER IN THE WORLD HAS THE MOST ADVANCED SOURCE LEVEL DEBUGGER IN THE WORLD:

Manx Aztec SDB.



- View your C Source
- Enter commands
- View your command output ...

... all at the same time!

Save Time and Effort

If you're a pro at working with low level Amiga debuggers, you'll recognize what a source level debugger can mean—*time savings*. Time you'd rather spend *creating* than debugging. And if you're a beginner, *SDB* will make you a pro in no time.

Outstanding Features

That's why our new windowed *SDB* is so spectacular—because it's full of exciting features that make debugging a breeze. Of course, *SDB* has all of the features you expect from a debugger—like line-by-line tracing. Conditional breakpoints on lines, functions, or variables. Examination, modification, and display of global, local, and static variables, structures or expressions by name.

But *SDB* is also full of unexpected, incredibly sophisticated features. There's reusable command macros and procedures. Back tracing. Active frame context switching—just to name a few. Wait till you see *SDB* in action—it will blow you away!

Our Commitment to You

Manx Software Systems is the leading edge in C development systems. That means continual updates that bring the best to you.

Experience version 3.6 of Aztec C68k/Am today. And enjoy the most advanced debugger available ... on *any* computer.

AZTEC C VERSIONS TO SUIT YOUR NEEDS:

Aztec C68K/Am-p Professional System $199
Includes all of the Standard Features

Aztec C68K/Am-d Developer System $299
Includes all of the Standard and Extended Features

Source Level Debugger. $75

Library Source $300

Standard Features of Aztec C68k/Am 3.6:

- optimized C with selectable 68020 and 68881 support
- 680x0 Macro Assembler with 68881 support
- linker/librarian with overlays, scatter load, and segmentation
- symbolic debugger
- supports Amiga object format
- UNIX, AMIGA, and general purpose run time routines
- runs under CLI and supports all Workbench functions
- creates CLI and Workbench applications
- Supports both 1.1 and 1.2 Amiga DOS
- 600 pages of documentation and great example programs

Extended Features of Aztec C68k/Am 3.6

- UNIX utilities make, diff, grep, obj, ord, and vi
- special math support libraries for 68881 and Manx IEEE emulation

Portability: Aztec C is also available for the Macintosh, Apple II, MS-DOS/PC-DOS, CP/M-86, TRS-80, ROM, and others. Aztec C68k Third Party Software: A large array of support software is available for Aztec C86k. Call or write for information. The following is a list of the most requested products: **Power Windows • Amiga View • Key to C • Amiga Lint • Metascope.** Immediate Delivery Available for Most Destinations. Aztec C is available on a thirty day money-back guarantee. **C.O.D., VISA, American Express, MasterCard, wire (domestic and international),** and terms are available.

Order Now At No Risk Or Call For Information

1-800-221-0440

In NJ or outside the USA call: 201-542-2121
Telex: 4995812MANX Fax: 201-542-8386

MANX

Manx Software Systems • One Industrial Way • Eatontown, NJ 07724

SEE FOR YOURSELF WHAT ALL THE FUSS IS ABOUT ...

Order our Manx Aztec C68k/Am with 30-day satisfaction guarantee. We're convinced that once you see *SDB* at work, no other debugger will ever be good enough again. But if you don't believe us—try us! We're offering an SDB Demonstration Disk for just $5. Simply call 1-800-221-0440 (NJ call 201-542-2121) and order your copy of Aztec C or your Demo Disk today.

alternates them as part of a defined sequence. If you are typing in these programs, notice that some of the structure definitions have changed from the first program. Because of the complex linking of the animation structures, and because of the lack of forward referencing in the available C compilers for the Amiga, it is a real challenge to find an easy way to initialize all these structures without creating a tangle. There are other ways to do it, but you can at least see clearly how the variables in the structures are declared.

```
 /*   Program Two-sequenced animation.
*/
/*   The notes precede the relevant
*/
/*   statements and are explained
*/
/*   after the program listing
*/

#include <functions.h>   /* for Manx
                            only */
#include  <intuition/intuition.h>
#include  <graphics/gels.h>
struct  IntuitionBase  *IntuitionBase;
struct  GfxBase  *GfxBase;
struct  Screen  *Screen1;
struct  Window  *Window1;
struct  ViewPort  *WVP1;
    /***   NOTE 1 ***/
struct  AnimOb  Obj1,  *animKey;
struct  AnimComp  comp1,  comp2;
struct  Bob  b1,b2;
struct  VSprite  s1,s2;
struct  GelsInfo  gelsinfo;
struct  collTable  Ctable;
#define  RP1   Window1->RPort
VOID  Drawit();

WORD  Image_data1[130]  =  {
   /*  Width:      18  (pixels)
       Height:    13  (pixels)
       Depth:      5   (planes)*/
   0x0000,0x0000,0x0000,0x0000,0x0080,
   0x0000,0x0080,0x0000,0x0080,0x0000,
   0x0080,0x0000,0x0080,0x0000,0x0080,
   0x0000,0x0080,0x0000,0x0080,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0xffff,0x8000,0xffff,0x8000,
   0xfc1f,0x8000,0xfc1f,0x8000,0xfc1f,
   0x8000,0xfc1f,0x8000,0xfc1f,0x8000,
   0xfc1f,0x8000,0xfc1f,0x8000,0xfc1f,
   0x8000,0xffff,0x8000,0xffff,0x8000,
   0xffff,0x8000,0x0000,0x0000,0x0000,
   0x0000,0x03e0,0x0000,0x03e0,0x0000,
   0x03e0,0x0000,0x03e0,0x0000,0x03e0,
   0x0000,0x03e0,0x0000,0x03e0,0x0000,
   0x03e0,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0360,0x0000,0x0360,
   0x0000,0x0360,0x0000,0x0360,0x0000,
   0x0360,0x0000,0x0360,0x0000,0x0360,
   0x0000,0x0360,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000  };

WORD  Image_data2[130]  =  {
/*     Width:      18
       Height:    13
       Depth:      5   */
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0xffff,0x8000,0xffff,
   0x8000,0xffff,0x8000,0xfc7f,0x8000,
   0xfbbf,0x8000,0xf7df,0x8000,0xffbf,
   0x8000,0xff7f,0x8000,0xfeff,0x8000,
   0xfdff,0x8000,0xfbff,0x8000,0xf00f,
   0x8000,0xffff,0x8000,0xffff,0x8000,
   0xffff,0x8000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0xffff,
   0x8000,0xffff,0x8000,0x0000,0x0000,
   0x0380,0x0000,0x0440,0x0000,0x0820,
   0x0000,0x0040,0x0000,0x0080,0x0000,
   0x0100,0x0000,0x0200,0x0000,0x0400,
   0x0000,0x0ff0,0x0000,0x0000,0x0000,
   0xffff,0x8000,0xffff,0x8000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0x0000,0x0000,0x0000,0x0000,
   0x0000,0xffff,0x8000,0xffff,0x8000  };

USHORT  colormap[32]  =  {
   0x0ccd,0x0585,0x0f79,0x0f30,0x0f9b,
   0x0f90,0x0fc3,0x0fc9,0x0eef,0x0ddd,
   0x0ccc,0x0aaa,0x0fdc,0x0fcb,0x0fba,
   0x0ea9,0x0e93,0x0d90,0x0c87,0x0a54,
   0x0c75,0x0f67,0x0555,0x0069,0x09e1,
   0x0fed,0x07c1,0x05a0,0x0270,0x0fdd,
   0x0fff,0x0000   };

WORD  Sbuffer1[2  *  13*  5];
WORD  Cmask1[2  *  13];
WORD  Bline1[2];

WORD  Sbuffer2[2  *  13  *  5];
WORD  Cmask2[2  *  13];
WORD  Bline2[2];


struct  NewScreen  NewScreen1  ={
    0,0,320,200,5,1,0,
    NULL,CUSTOMSCREEN,
    NULL,NULL,NULL,NULL   };

struct  NewWindow  NewWindow1  ={
    0,0,320,200,1,0,CLOSEWINDOW,
    SMART_REFRESH  |  ACTIVATE  |
    BORDERLESS  |  WINDOWCLOSE,
    NULL,NULL,NULL,NULL,NULL,0,0,0,
    0,CUSTOMSCREEN   };

struct  VSprite  v1  ={
    NULL,NULL,NULL,NULL,NULL,NULL,
    OVERLAY  |  SAVEBACK,
    0,0,13,2,5,0,0,&Image_data1[0],
    &Bline1[0],&Cmask1[0],NULL,&b1,
    0x01f,0,NULL  };

struct  VSprite  v2  ={
    NULL,NULL,NULL,NULL,NULL,NULL,
    OVERLAY  |  SAVEBACK,
    0,0,13,2,5,0,0,&Image_data2[0],
    &Bline2[0],&Cmask2[0],NULL,&b2,
    0x01f,0,NULL  };

main()
{
Open_Libraries();

Open_Screens();

Init_Bobs();

Drawit();

Cleanup();

}     /***  end  main  ***/

Open_Libraries()
   {
    IntuitionBase  =  (struct
     IntuitionBase  *)
     OpenLibrary("intuition.library",0);

    GfxBase  =  (struct  GfxBase  *)
     OpenLibrary("graphics.library",0);
   return();
   }

Open_Screens()
  {
    Screen1=OpenScreen(&NewScreen1);
    NewWindow1.Screen=Screen1;
    Window1=OpenWindow(&NewWindow1);
    WVP1  =  (struct  ViewPort*)
     ViewPortAddress(Window1);
    LoadRGB4(WVP1,&colormap,32);
   return();
  }

  Init_Bobs()
  {
       /*** NOTE 2 ***/
    b1.Flags=BOBISCOMP;
    b1.SaveBuffer=&Sbuffer1[0];
    b1.ImageShadow=&Cmask1[0];
    b1.BobVSprite=&v1;
    b1.BobComp=&comp1;
    b1.DBuffer=NULL;

    b2.Flags=BOBISCOMP;
    b2.SaveBuffer=&Sbuffer2[0];
    b2.ImageShadow=&Cmask2[0];
    b2.BobVSprite=&v2;
    b2.BobComp=&comp2;
    b2.DBuffer=NULL;

       /*** NOTE 3 ***/
   Obj1.NextOb=NULL;
   Obj1.PrevOb=NULL;
   Obj1.AnY=64*20;
   Obj1.AnX=64*10;
   Obj1.YVel=0;
   Obj1.XVel=0;
   Obj1.YAccel=0;
   Obj1.XAccel=0;
   Obj1.RingYTrans=64*1;
   Obj1.RingXTrans=64*2;
   Obj1.HeadComp=&comp1;

       /*** NOTE 4 ***/
   comp1.Flags=RINGTRIGGER;
   comp1.Timer=30;
   comp1.TimeSet=30;
   comp1.NextComp=NULL;
   comp1.PrevComp=NULL;
   comp1.NextSeq=&comp2;
   comp1.PrevSeq=&comp2;
   comp1.AnimCRoutine=NULL;
   comp1.YTrans=0;
```

```
    comp1.XTrans=0;
    comp1.HeadOb=&Obj1;
    comp1.AnimBob=&b1;

    comp2.Flags=RINGTRIGGER;
    comp2.Timer=50;
    comp2.TimeSet=50;
    comp2.NextComp=NULL;
    comp2.PrevComp=NULL;
    comp2.NextSeq=&comp1;
    comp2.PrevSeq=&comp1;
    comp2.AnimCRoutine=NULL;
    comp2.YTrans=0;
    comp2.XTrans=0;
    comp2.HeadOb=&Obj1;
    comp2.AnimBob=&b2;

    gelsinfo.nextLine = NULL;
    gelsinfo.lastColor = NULL;
    gelsinfo.collHandler =NULL;
    RP1->GelsInfo = &gelsinfo;
    InitGels(&s1, &s2, &gelsinfo);
        /*** NOTE 5 ***/
    GetGBuffers(&Obj1,RP1,NULL);
    InitGMasks(&Obj1);
    InitAnimate(&animKey);
    AddAnimOb(&Obj1,&animKey,RP1);

    return();
}


VOID  Drawit()
  {
   SHORT x;
     for (x=0; x<=700; x++)
     { Animate(&animKey,RP1);
       SortGList(RP1);
       WaitTOF();
       DrawGList(RP1,WVP1);
     }
  }
Cleanup()
 {
   Wait(1<<Window1->UserPort->mp_SigBit);
   FreeGBuffers(&Obj1,RP1,NULL);
   CloseWindow(Window1);
   CloseScreen(Screen1);
   CloseLibrary(GfxBase);
   CloseLibrary(IntuitionBase);
  return();
 }
```

NOTE 1

These next 4 structures are the major additions and changes from the first program. If you are typing these programs in by hand, the first bob definition is unchanged from the first program, but we've added a different second bob and VSprite.

NOTE 2

In the bob definition we change two parameters to show that they are now part of an Animation Component, and to link them with their respective AnimComps.

NOTE 3

The first two flags are manipulated by the system. Alter these only if you want to unlink an AnimOb from the animation list. All of the numeric data in this structure is supposed to be a binary fraction.

NOTE 4

The RINGTRIGGER flag is set in both AnimComps since our sequence has only 2 views. If your sequence had more you would set the RINGTRIGGER flag only in the last view and the rest would be NULL. This flag signals the controlling AnimOb to change the registration point of the whole sequence by the amount you have specified in the RingXtrans and RingYTrans variables. The rest of the parameters are explained in the RKM; play with these if you wish.

NOTE 5

The first two statements allocate the buffers and masks for the bobs; they are different from the statements for simple bobs. Notice there is a mirror statement (FreeGBuffers) to deallocate the buffers. The next two statements initialize the animation system and add your AnimOb to the system list. If you were going to create a second AnimOb you would add a statement like AddAnimOb(&Obj2, &Obj1, RP1).

We can't put off discussing double buffering any longer, so the next installment will tackle this issue, and we will create the ultimate animation example—double buffered sequence and motion animation. It really isn't hard, especially after someone has shown you how to do it, but it is cumbersome. It also uses precious memory, since it involves creating 2 bitmaps for each screen and another save buffer for each bob. I hope the 2 Meg graphics chip did not get buried at Los Gatos, as the current 512K limit for graphics and sound data is a major weakness of the Amiga.

•AC•

AmigaNotes

Rick Reviews Four ***Digital Music Reference Books***

by Rick Rae
CIS (76703,4253)

Boy, time flies when you're having fun with new toys. I've been playing with three new audio oriented packages, and it seems as if I just sent in a column last week. Two of the packages are pre-release with hand written disk labels, and the manual for the third is clearly stamped "Beta", so I can't do more than hint about them at the moment. But trust me: interesting stuff is coming down the pike!

Ah well, the new toys will just have to go on the back burner for a while, because I want to talk resources for a bit. I've gotten lots of inquiries about reference books from other AmigaForum members on CompuServe, so I expect there are a few Amazing readers with the same sort of questions. Anyone who is seriously interested in programming the Amiga knows about the ROM Kernel Manuals and other materials; what I'll be talking about are instead aimed squarely at audio and synthetic sounds.

DIGITAL AUDIO SIGNAL PROCESSING: AN ANTHOLOGY
— John Strawn, Editor
$34.95 William Kaufmann, Inc.

This book is one of several currently available in the **Computer Music and Digital Audio** series. The intent, says the publisher, is to provide a central source for books which deal with computer music, digital audio, and related subjects.

To delve into digital signal processing, you need a good grasp of mathematics. Some things can be presented without the "higher levels" of mathematics such as calculus, but a basic understanding of algebra, trigonometry, and the like is an absolute necessity. Keeping this in mind, Strawn has selected an excellent first chapter: **An Introduction to the Mathematics of Digital Signal Processing**, by F. Richard Moore.

Moore acknowledges that it has been quite a while since high school or college for some of us, and presents a tutorial on the math skills needed to comprehend the rest of the book. The chapter begins with an introduction to algebra by explaining the meaning of "y=x+1" (you can't get much simpler than that) and proceeds through trigonometry, sampling, transforms, and digital filtering. At the end of each section the author has provided a set of problems to test the reader's understanding; the only "gotcha" here is that no answers are included.

The second chapter is **An Introduction to Digital Filter Theory**, by Julius O. Smith. This section begins with a brief overview and shows how a digital filter can be represented by the statement "y[n]=x[n]+x[n-1]" within a programmed loop. From here, Smith delves into deeper topics, such as theoretical foundations and special cases. As before, problems (but no solutions) are provided.

"Spiral formations are known to occur in the structure of proteins and in the great arms of the Andromeda nebula. Spirals twist their way through an amazing variety of things, including seashells, plants, and DNA. Because of the structure of their eyes, even certain insects follow the path of a logarithmic spiral as they make their way into the candle flame..."

With this intriguing opener, Tracy Lind Petersen begins the third chapter on **Spiral Synthesis**, a relatively new way of looking at synthesis. Although the shortest article in the book, its 11 pages are packed with intriguing information about this technique, including a two page demonstration program in C.

The fourth chapter, by James A. Moorer, is titled **Signal Processing Aspects of Computer Music: A Survey**. Newcomers to music synthesis may well find this the most valuable section of the book, and I would urge you to read it immediately after the first chapter, if not before. Mr. Moorer does an excellent job of laying the groundwork of additive, subtractive, and nonlinear synthesis. The mathematics used are minimal, and if you understand high school math you should have little difficulty. After covering basic synthesis approaches, Mr. Moorer goes on to discuss simulation of room acoustics and instrument placement and, briefly, the concepts behind the all-digital recording studio.

The final chapter, **An Introduction to the Phase Vocoder** by John W. Gordon and John Strawn, is a bit specialized but interesting nonetheless. The vocoder (VOice CODER) was originally a research effort aimed at reducing the bandwidth needed to send speech over phone lines. Al-

though the vocoder was not successful in its intended area, someone tinkering with it recognized the musical potential, and the research took a sudden left turn.

Stated briefly, a vocoder slices the audio frequency spectrum into numerous bands of parametric information, then builds a new signal based on the composite of all the input data. If no changes are made during the reassembly, you get out what you put in. By substituting new signals for the reassembly, almost anything is possible.

Gordon and Strawn provide sample vocoder routines written in, unfortunately, SAIL (a superset of ALGOL). Happily, the code is well commented, and SAIL is straightforward enough that an interested programmer should have little difficulty converting to whatever language might be handy.

DIGITAL AUDIO ENGINEERING: AN ANTHOLOGY — John Strawn, Editor
$29.95 William Kaufmann, Inc.
The first chapter is **An Introduction to Digital Recording and Reproduction** by James F. McGill. After a brief historical overview, McGill explains the concepts behind digital recording, including the limitations of quantization and the Nyquist theorem. From here, he departs from all the other references I normally use. Rather than going on about digital systems in general, he provides a brief but excellent treatment of Pulse Code Modulation (PCM) as applied to digital recording, supported by sections on input and output conditioning. If you've ever wanted to know what goes on inside your CD player or a digital mixing console, this chapter will give you a good idea.

Limitations on the Dynamic Range of Digitized Audio, by Robert Talambiras, desperately needs to be read by some Amiga programmers. Some of us have heard promises of 90 db S/N ratios with 8 bit samples; those of us who have listened to samples from the Amiga know better. This chapter explains, in reasonable detail, the difference between 84 db of dynamic range (which we **can** achieve) and the Amiga's 48 db S/N ratio, and why both limitations exist.

The third chapter is **Architectural Issues in the Design of the System Concepts Digital Synthesizer**, by Peter R. Sampson. This is an intriguing discussion of one of the first large digital synthesizers, installed at Stanford University. Owners of the DX7 (which has six "oscillators" per voice and sixteen voices) will have a good feel for the size of this system, which include **256** digital oscillators and is built up from approximately 2500 integrated circuits! To any reader who is surprised by the size of the system (I was), I will offer this teaser from Mr. Samson: "The question is often asked why this synthesizer is so big. A better question would be why it is so small."

The FRMbox – a Modular Digital Music Synthesizer, by F. Richard Moore, is a brief discussion of a much smaller experimental system. Designed to be a flexible "workbench" for digital audio experiments, the FRMbox might be a worthwhile case study for programmers toying with digital synthesis on the Amiga.

With the Amiga getting into video and audio throughout the television, radio, and film industries, more and more home computer enthusiasts are becoming interested in what the "big boys" do, and how they do it. Chapter five, by James A. Moorer, is entitled **The Lucasfilm Digital Audio Facility**, and is a sort of "guided tour" through the requirements and design of that system. Although special hardware was required for the digital signal processing (phrases like "64 channels of 50 khz digital audio" float through the article), it's interesting to note that the control and console computers were based on Motorola 68000s. The control computer, in fact, is built around a modified SUN Systems CPU.

THE FFT: FUNDAMENTALS AND CONCEPTS by Robert W. Ramirez
Prentice-Hall, Inc.
Fourier theory has been developing in spurts ever since the concept was first put to paper in the early 1800s. The 1960s brought us the Fast Fourier Transform, a new approach which substantially reduced calculation overhead. The same principles applied to sampled data produce the Discrete Fourier Transform. Recently, the Hartley Transform, hinting at a 50% reduction in calculation time, has been receiving press. Ramirez's book focuses on the FFT and DFT.

The Fourier theory allows us to move freely between the time and frequency domains. In other words, we can analyze the waveform of an instrument and determine its harmonic spectrum, or combine a set of frequencies to produce a corresponding waveform. The FFT is a cornerstone of many synthesis techniques, yet it is often used without a real understanding of how it works. This book does a remarkable job of clarifying the theories behind the FFT; I will go so far as to say it is the best explanation I've read, and the one to which I turn when things get a bit fuzzy.

Ramirez's effort is divided into three sections. The first is an introduction to Fourier theory, and an explanation of the Fourier Integral and Series. The tutorial is well illustrated and complicated math is kept to a minimum. If you understand trigonometry and the simplest fundamentals of integration and summation, the discussion will be understandable.

The second section deals less with theory and more with actual application of the FFT and DFT. After explaining the mechanics of the transform, the discussion moves on to

continued...

potential problems of a digital system: timing jitter, quantization error, leakage, periodicity problems, and aliasing.

The final section includes a description of the Sande-Tukey DFT algorithm, as well as an MBASIC implementation of the FFT. The author does a good job of explaining the operation of the program and pointing out potential problem spots for those interested in converting this sample code to other varieties of BASIC. Other practical concerns, such as interpreting and improving the results of a transform, are also covered.

MUSICAL APPLICATIONS OF MICROPROCESSORS
by Hal Chamberlin
$39.95 Howard W. Sams Company

This work is, without question, the "hardware hacker's bible" when it comes to electronic music synthesis, and rightly so: Hal Chamberlin has been involved with musical micros just about as long as there have been micros.

If memory serves, I first met Hal Chamberlin in Gathersburg, Maryland at a amateur radio festival, or "hamfest". He was demonstrating a homebrew 8008 tied to a paper tape reader, printer, and a surplus vector driven radar display (all "state of the art" at the time!). Even at that point, more than ten years ago, he was working with programs which allowed him to design sounds using additive synthesis, draw the resulting waveform on the radar display, and simultaneously play the tone.

Since that time I've run into Hal on a regular basis at the larger Eastern computer shows and hamfests. At one of the major shows in Philadelphia he demonstrated his non-realtime synthesis system by playing back, from 8" floppies, his rendition of Bach's Toccata and Fugue in D Minor; the synthetic timbre was hauntingly close to that of a huge pipe organ. A few years later, at the Trenton Computer Festival, he was demonstrating realtime sampling to floppy by recording and playing back selections from an LP.

It's highly unlikely that Hal Chamberlin would remember me, as I was just a persistent face in the crowd. But Hal, and his achievements, are hard to forget. He was one of the first people to demonstrate what could be done in the audio field with a simple 8 bit CPU, and his book is somewhat of a compendium of his knowledge in the field.

This large book is divided into four sections. The first section is packed with background information on music synthesis, tape manipulation, voltage controlled analog synthesizers, and microprocessors. This section serves as an excellent introduction for those readers who have never had the opportunity to deal with the older machines.

The second section deals with a hybrid approach to synthesis: using a microprocessor to control dedicated analog hardware. Covered here are the basics of A/D and D/A conversion, methods of computer control, and an introduction to display systems.

The third section launches into pure digital approaches, and it is possibly here an Amiga owner would spend the most time. Hal explains how to increase the dynamic range and reduce the distortion of A/D/A systems, then begins a discussion of various approaches to digital sound synthesis. One of the methods described—the table lookup approach—is very similar to the way the Amiga's custom hardware generates sound. This section also covers digital filters, reverberation, chorusing, percussive sounds, spectral analysis, FFTs, and a number of digital synthesis systems.

The final section is an overview of the current "state of the art" in commercial synthesizers. Or, rather, state of the art at the time the book was printed: electronic music hardware is changing so rapidly only a magazine can truly keep up with the cutting edge. Nevertheless, the book takes a look at such machines as the Synclavier, Kurzweil, and Fairlight.

Hal closes the book with some predictions. How close has he come so far? Remembering that electronic music is a highly volatile and changeable field, Hal obviously has either a firm understanding of where we've been and where we're going, or has a reasonably accurate crystal ball:

"... 8-bit [CPUs] will descend to the status of current 4-bit devices... 16– bit microprocessors... will be used in... low-end computers, while the 32–bitters will be... for business-oriented personal computers... look for portable synthesizers using digital playback of recorded sounds as their 'synthesis' technique to become available for under $1,000... even cheap toy instruments (under $100) will have outstanding sonic capability... but with cheap keyboards and cases... the typical home personal computer will have at least an eight-voice synthesizer as standard equipment, again using prerecorded sounds..."

The only thing which keeps me from recommending this book to anyone interested in making electronic music is the wide ground it covers. If you are interested primarily in using the Amiga's hardware, or building an analog synthesizer, or writing some analysis programs, much of this book will be a waste. If, however, you want a reference book which covers the majority of the field, this book should most definitely be on your shelves.

Nybbles, Rick

•AC•

Dream on...

THE DIRECTOR™

Envision a creative freedom you've only *dreamed* about... building animations, presentations and computer films that use the *full* power of the Amiga. Imagine page flipping, color cycling, text generation, even Videoscape 3D™ animations, all *combined* at the same time on the same screen. Until now this has been just a dream. Now the dream comes true with the Director.

Combine pictures in any resolution from any paint program or digitizer you use with your Amiga. With the Director you write a script to display them in unlimited ways. You can page flip full or partial screens, use drawing commands, create special effects, generate multiple font text, execute AmigaDOS commands, and much more. This is only the beginning of the freedom you give yourself with the Director.

From the simplest slideshow to the most sophisticated desktop video, the Director will help turn your dreams into reality.

Professional display and animation software for the Amiga™

Other features include:

- Effects: Fades, Dissolves, Wipes, Blits, Stencils
- Digitized soundtrack module
- Preload images, fonts, and sounds up to your memory limit
- Supports HAM and overscan
- Keyboard and mouse interaction
- Random number generator
- Text string and file input and output
- No copy protection

And more...

$69.95

Check or money order payable to:
Right Answers

Plus $3 shipping and handling
Calif. residents add 6% sales tax

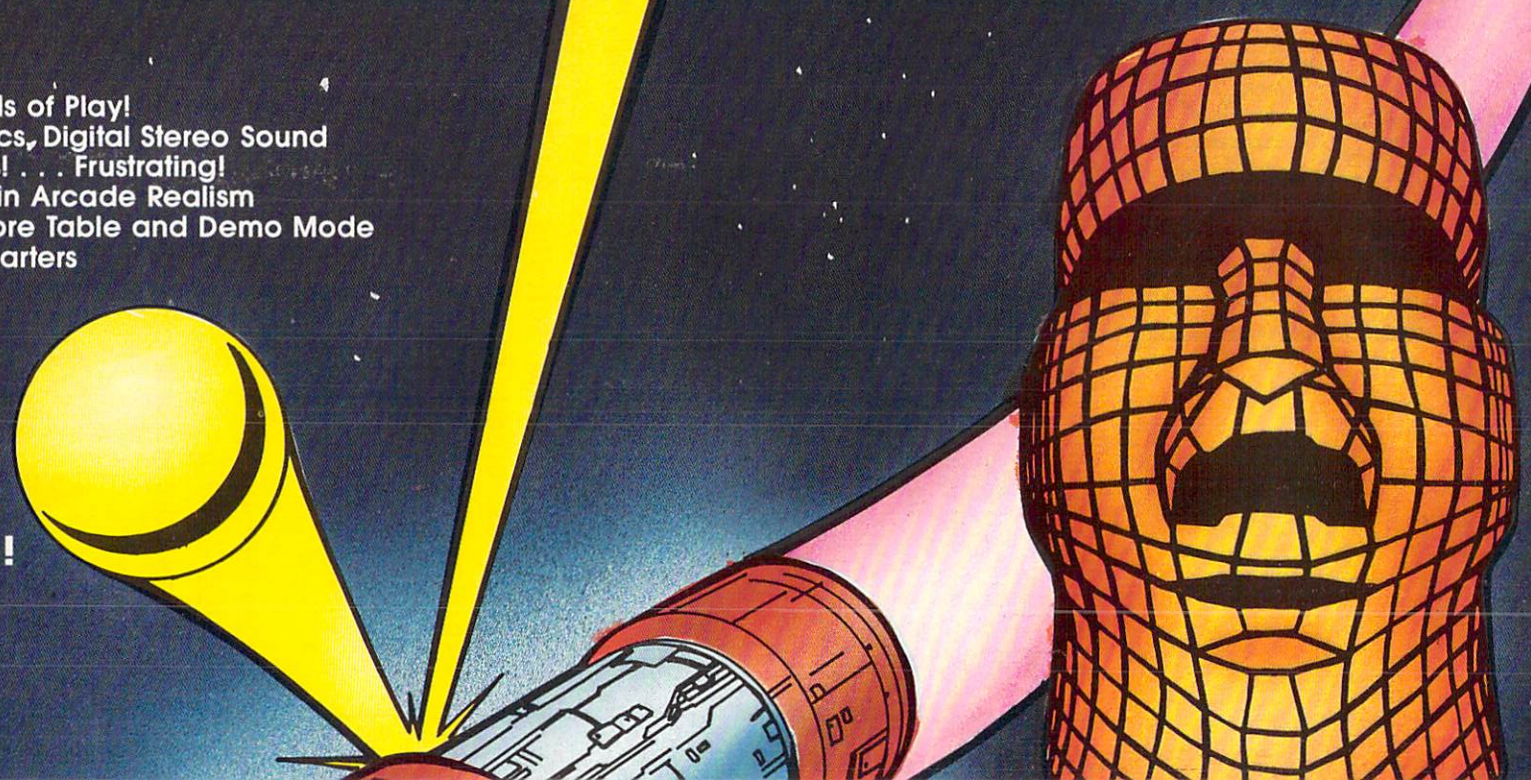
The Right Answers Group
Department C
Box 3699
Torrance, CA 90510
(213) 325-1311

Amiga is a trademark of Commodore-Amiga, Inc. Videoscape 3D is a trademark of Aegis Development, Inc.

ARKANOID™

THE #1 ARCADE BLOCK-BUSTER!

- **33** Sizzling Levels of Play!
- Dazzling Graphics, Digital Stereo Sound
- Fast! Fun! Furious! . . . Frustrating!
- A Breakthrough in Arcade Realism
- Arcade High Score Table and Demo Mode
- Never Needs Quarters



ONLY $49.95 * !

SIMPLY THE BEST AMIGA GAME EVER!

MARAUDER

"Marauder II is an excellent utility at a bargain price."
Bob Ryan-Amiga World

"Every Amiga owner should have a copy . . ."
Bruce Webster-BYTE

"If you own an Amiga and don't have Marauder II from Discovery, you don't have one of the best tools an Amiga owner can have." -INFO Magazine

The best copier for Amiga™ at any price! Still only **$39.95** *

NEW! DX SERIES

Our new **DX SERIES**™ provides the power and features of the world renowned Hewlett-Packard advanced programmable calculators at a fraction of the price. The **DX-16**™ is a perfect on screen replica of the HP-16C™, the ultimate programmer's calculator. The **DX-11**™ mirrors the HP-11C™, an excellent choice for students, engineers and general purpose calculations. Both are included to make this package a double value. Both are fully programmable and fully multi-tasking power tools. Complete tutorials and excellent documentation will get you started quickly. **$49.95** *

NEW! AMNIX

AMNIX™ delivers the finest features of the famous UNIX™ C-Shell. Free yourself from constant disk swapping. Over 40 high-speed resident commands make their disk-based Amiga DOS counterparts obsolete. **AMNIX**™ also provides extended batch files, command line history and editing and environment variables. It's easy to learn and superbly documented. A "must have" for Amiga power users! **$49.95** *

Amiga™ is the trademark of Comodore-Amiga, Inc., UNIX™ is the trademark of Bell Laboratories Inc., HP-16C™ and HP-11C™ are trademarks of Hewlett Packard Co.

LIMITED TIME OFFER!

ORDER DIRECT AND GET
$5.00 OFF
EACH PRODUCT YOU BUY
PLUS

FEDERAL EXPRESS

ABSOLUTELY FREE!!

TO ORDER CALL TOLL FREE
1-800-34-AMIGA
or (301) 268-9877

or send check, cash or
VISA or MasterCard numbers to:
Discovery Software International Inc.
163 Conduit Street
Annapolis, MD 21401

*Order now for $5 introductory discount!

(Federal Express shipping in continental U.S. only.)

VISA OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Basic Text

What if you want to position your text to a specific pixel location which doesn't correspond to a row/column location? EASY!!!

by Bryan Catley

The first Basic command we learn is almost always PRINT . . .and with good reason. With this one command (and a couple of others), a great many simple programs can be created! However, before long, this knowledge is expanded to include the appropriate method of positioning the printed text at any desired location on the screen. With the Amiga, this is the "LOCATE row,column" command.

This command allows you to position text at any row/column location on the screen. Note that the number of rows is always 24 or 48, but the number of columns may vary from 30 to 80, both depending upon the Preferences settings and the current screen definition. The main point of all this is that you are restricted to specific row and column locations. What if you want to position your text to a specific pixel location which doesn't correspond to a row/column location? Well, the Amiga is one of the few computers which provides you with this capability!

Horizontal Positioning by Pixel

You are probably already familiar with TAB(n). Regardless of the current column, the next item will be printed "n" columns from the left side of the screen, even if it overlays some of the text previously printed in that row.

Well, PTAB(n) functions in exactly the same way as TAB(n), except it works with pixels, rather than with columns. Try the following program:

```
FOR n=0 TO 12
  LOCATE 5+n,1
  PRINT PTAB(10+n);"See where we are!"
NEXT
```

As you can see, positioning text horizontally by pixel is really very simple! Vertical positioning by pixel is, however, a little more involved. Unfortunately, there is no AmigaBasic command which allows you to vertically position text by pixel. You must use the Move& operating system routine. However, it really isn't that difficult, and it has the added advantage of also allowing you to horizontally position text by pixel at the same time! What this boils down to is that if you use Move&, you no longer have to use LOCATE or PTAB(n); although you may if you wish.

Move& sets the next print position to the given pixel x/y position. For example: CALL Move&(WINDOW(8),30,50) will cause the next PRINTed items to be located 30 pixels from the left, and 50 pixels down from the top of the current window. Which piece of the first character is placed at the given x/y position, though? If you guessed the upper left hand corner, you're wrong! How about the lower left hand corner? Wrong again! One of the right hand corners? Still wrong!

To understand the correct answer, you must first understand that each character printed on the screen occupies a space of eight pixels by eight pixels, with the eighth, or bottom row of pixels, being blank. However, the "tails" of certain letters, such as "g" and "y", do extend into this row. The point of all this is that the seventh pixel down is the lower level of most letters. This point, known as the "baseline," provides the answer to the original question! Move& specifies the left side of the baseline for the next character to be printed!

Now, before we can see an example, there is one more thing to understand. Whenever we use an operating system routine, we must also specify which library it belongs to. In the case of Move&, it's the graphics library and it must be specified as LIBRARY "graphics.library" at some point before the first Move& is executed (Why the graphics library? Unlike many computers, the Amiga has no separate text mode. In fact, it treats text as just another form of graphics!).

But wait, there is still one more thing you must be aware of. In addition to specifying the library, it is also your responsibility to make sure there is a matching bmap file in the current directory! This means that if we specify "graphics.library," there must also be a "graphics.bmap" file directly available to Basic. Now, not all .bmap files are readily available, but there is a "graphics.bmap" file in the BasicDemos draw of your AmigaBasic disk. The trouble is, the BasicDemos draw will almost certainly not be the current directory, so you will have to change the directory as well! This is easy, so let's see an example of using Move& to position text both vertically and horizontally.

```
' Set correct directory path
CHDIR ":BasicDemos"
LIBRARY "graphics.library"
' Reset directory
CHDIR ":"
m&=35
FOR n&=20 TO 80 STEP 6
  CALL Move&(WINDOW(8),n&,m&)
  PRINT "Hello!"
  m&=m&+1
NEXT
```

Great! Now you know how to position text at ANY location on the screen. One of the great things about including a LIBRARY "graphics.library" statement in the program is that we now have access to every routine in the graphics library without worrying about defining the library again! This feature is very good as far as text is concerned because there are a number of routines we will find very useful.

Different Text Styles

Once you have the power of the graphics library routines at your fingertips, one of the first things you'll want to do is to change the style of your text output. By using the operating system routine SetSoftStyle&, you canchange the output text style from the "normal" style to italics, bold, or underlined. Besides the four basic styles, you may also combine them to produce combinations such as bold italics; underlined, bold "normal" text, etc. This task may be accomplished with the statement:

CALL SetSoftStyle& (WINDOW(8),Style%,255).

WINDOW(8), in case you hadn't realized it, provides Basic with a pointer to information about the current output window. The value assigned to Style% indicates the desired style. The 255 indicates that all styles are valid with the current font (This is not necessarily true if you use various disk based fonts, but for the purposes of this article, we will consider it to always be true).

Style% may assume any of the following values:

0 -plain (or "normal") text
1 -underlined text
2 -bold text
3 -underlined, and bold text
4 -italics text
5 -underlined italics
6 -bold italics
7 -underlined, and bold italics

This is great to be able to just set a new text style, but what if we want to turn the various text styles on and off as instructed by the program user (for example)? If we assume the variable Style% contains the current text style, then we may turn the various styles on and off with the following statements:

```
Turn on plain text:  Style%=Style% AND 3
Turn on underline:   Style%=Style% OR 1
Turn off underline:  Style%=Style% AND 6
Turn on Bold:        Style%=Style% OR 2
Turn off bold:       Style%=Style% AND 5
Turn on italics:     Style%=Style% OR 4
```

If you don't understand the ORs and ANDs, don't worry. Just be aware that selecting italics automatically turns off plain, and selecting plain automatically turns off italics. Underlined and bold may be turned on and off at will, regardless of whether plain or italics has been set.

Now is a good time to see the result of using the PRINT statement with a text style other than "normal." Enter and run the following short program. Forget the comment lines if you like:

```
CHDIR":BasicDemos"
LIBRARY "graphics.library"
CHDIR":"
PRINT "This is normal text."
' Set italics text style
CALL SetSoftStyle&(WINDOW(8),4,255)
PRINT "This is italicized text."
' Reset to normal text before quiting
CALL SetSoftStyle&(WINDOW(8),0,255)
END
```

What is the result? The italicized text is almost unreadable. You have just discovered a severe limitation of the PRINT statement!

We've already discussed how each character printed on the screen occupies an eight by eight pixel area. So, keeping this in mind, let's look a little closer at what happened when we "PRINTed" some italics.

Italics are formed by leaning each character to the right, but, to do this, each character must lean into the area occupied by the character to its immediate right. So, how can each italics character fit into an eight by eight pixel area? It cannot! That's why the "PRINTed" italics are unreadable. The fact of the matter is that the PRINT statement prints each character individually. This means that each time a character extends into the next character's area, it gets chopped off at the edge of the eight by eight area! In turn, the italics are unreadable italics. Further, this approach to printing also chops bold text, but it is nowhere near as obvious.

Fortunately, the answer to this dilemma is very simple. The graphics library provides us with a Text& function which formats and prints (or, more correctly, draws) a given string of text. By working with a string of characters, rather than individual characters, this function is able to display the text exactly in the desired format.

continued...

The format of the Text& function is:

```
CALL Text&(WINDOW(8),SADD('string'),string-length).
```

Try replacing the second print statement in the above program example with the following statement:

```
CALL Text&(WINDOW(8),SADD('This is italicized text.'),24)
```

If you make this change, the italics will print perfectly! This is great, but there are two obvious drawbacks. First, it's a very cumbersome statement to use each time you want to print a piece of text. Second, its format is not very flexible.

Well, these two drawbacks may be easily resolved by creating a simple three line sub-program which may be included in any program which requires the Text& function. You may call the sub-program by any name you wish; but for our purposes, we'll use "Display." The sub-program might look like:

```
SUB Display (Txt$) STATIC
  CALL Text&(WINDOW(8),SADD(Txt$),LEN(Txt$))
END SUB
```

Once you have done this, you may include statements such as:

```
Display "This is italicized text."
Display Heading$
```

in any of your programs. They will now be correctly formatted and displayed, regardless of the selected style.

Before we look at a more complete example, there are a couple of things about Text& of which you should be aware:

• Text may be displayed at any selected location via the Basic LOCATE statement or the operating system routine Move&.

• Text& never forces a new line following the text display. This means you must always use LOCATE or Move& prior to each use of the text (Or you could follow each Text& call with a Basic PRINT statement such as: Display String$:PRINT" ". But remember, when using bold or italics, that final space will chop the last character "Displayed").

Let's look at an example which pulls all of this together. Enter Listing #1, SAVE it, and then RUN it. This little program shows you an example of each of the seven text display styles using both the PRINT statement and the Text& function.

Now to look at something a little different. Different Text Modes? Yes, there are two different text display modes you may choose from, along with two optional modifiers! The two modes are named JAM1 and JAM2 and the modifiers are COMPLEMENT and INVERSVID.

The standard drawing mode (which you've been using all along) is called JAM2. This rather strange name indicates that two colors are "jammed" into each of the eight by eight by eight pixel squares which are used to represent the drawn characters. The two colors are, you guessed it, the background and foreground colors! This may not be quite as obvious as it seems because most of the time, the background color of the screen is the same as the text background color. To get a better picture of what is really happening, enter the immediate command "COLOR 3,2" and then type something. If you are using the standard Workbench colors, this action will result in orange text on a black ground, while the screen color remains blue. Thus, both colors are "jammed" into place! (The immediate command COLOR 1,0 will return things to normal).

JAM1, as you have probably guessed, results in only one color being "jammed" into place; the foreground color. (The

background color is effectively ignored while JAM1 is being used). This drawing mode is very useful when you want to draw text over a graphics display, without disturbing ANY of the graphics.

To use JAM1, (and to later switch back to JAM2 and/or use the two modifiers), the SetDrMode& operating system routine is used as follows:

CALL SetDrMode&(WINDOW(8),Mode&)

Where Mode& assumes one of the following values:
0 Use JAM1 drawing mode
1 Use JAM2 drawing mode
2 Use JAM1 with COMPLEMENT modifier
3 Use JAM2 with COMPLEMENT modifier
4 Use JAM1 with INVERSVID modifier
5 Use JAM2 with INVERSVID modifier
6 Use JAM1 with COMPLEMENT and INVERSVID modifiers
7 Use JAM2 with COMPLEMENT and INVERSVID modifiers

In practice, modes 3 and 7 are rarely, if ever, used. We'll see why a little later.

Before we discuss the two modifiers, it is necessary to understand the Amiga's method of defining complementary colors. Every time you select a menu item, it changes color. Each time you click on an icon, it changes color. What determines what these new colors are to be? Well, each color (or palette) complementary color (or palette). This complementary color is used to show selected menu items and icons.

The determination of a complementary color is really quite simple. Just subtract the color (palette) number from the maximum color (palette) which is available on the current screen. The result is its complementary color. For example, on a 32 color screen, color 0's complementary color is 31; 1's is 30; 2's is 29; etc. On the four color Workbench screen, color 0's complement is 3; color 1's is 2; color 2's is 1; and color 3's is 0.

Keeping this in mind, we can now discuss the two modifiers!

Strange as it may seem, when COMPLEMENT is selected (Mode&=2, 3, or 7), neither the selected foreground or background color is used! Rather, the COMPLEMENTary color of the pixels which will be overwritten is used! For this reason, COMPLEMENT is not very useful with JAM2. Both foreground and background colors are "jammed" into the display area. This causes ALL the pixels in the character area to be complemented, resulting in a solid color display area! Thus, modes 3 and 7 are rarely, if ever, used.

INVERSVID is a little more simple, and as its name implies, it simply reverses the role of the foreground and background colors. However, one interesting aspect of all this is

that when used with JAM1, the background area surrounding the actual text is filled with the specified foreground color. The text itself is left untouched, allowing the original background color to show through!

All this may very well seem strange and somewhat difficult to understand. The best thing is to look at an example of each of the eight display modes. Enter Listing #2, save it, then execute it. If entered correctly, you will see a large circle filled with a random white-on-blue pattern on a gray background. Over this, eight lines of text are printed, one for each of the eight available modes. Yes, it really does looks strange! The first time I saw it I did a real double-take, but it is all correct, and the way it is supposed to be.

Let's take a closer look at each of the eight lines. Keep a copy of the program listing handy so you may easily determine color complements.

Line 1: JAM1.
The text is written in the foreground color (green), with no background color being "jammed" into the display area.

Line 2: JAM2.
The text is written in the specified foreground and background colors (green on red), which is the "normal" mode.

Line 3: JAM1/COMPLEMENT.
The text is written in the complementary colors of the display area it covers. Thus, it appears as white on the gray portion of the screen and as yellow and gray on the patterned area (Note the pattern is the same, just in its complementary colors!).

Line 4: JAM2/COMPLEMENT.
Asfar as the single color background is concerned, notice how all the over-written pixels are complemented, resulting in a single colored band (Remember, this mode causes the color of all over-written pixels to be complemented. When

continued...

they're all the same color, we end up with a single colored band). Notice also how the pattern area has been complemented. What was white is now gray, and what was blue is now yellow.

Line 5: JAM1/INVERSVID.
The specified foreground color (green) is jammed into the background area of each character, while the character itself is left untouched, resulting in a "see through" effect!

Line 6: JAM2/INVERSVID.
As you might expect, the specified foreground and background colors (green on red) have been reversed.

Line 7: JAM1/COMPLEMENT/INVERSVID.
Of the eight lines, this is probably the most interesting one. We have an inverted version of line #3, which makes for an interesting effect! Things start out the same as in line 3, but then everything is inversed. This results in gray-on-white text in the solid portion of the screen, and blue and white text on a yellow and black background in the patterned area!

Line 8: JAM2/COMPLEMENT/INVERSVID.
The result is similar to that from line 4. We end up with a single color band!

The program may be terminated by clicking the left mouse button.

Now, should you simply have a requirement for reversed colors (and nothing else), all you need is the AmigaBasic COLOR statement! Consider the following code fragment which flips colors one way, then flips them back:

```
PRINT"Normal JAM2 text.";
COLOR 0,1:PRINT" Inversed text. ";
COLOR 1,0:PRINT"Back to normal."
```

As a final note, if you have moved "graphics.bmap" out of the BasicDemos drawer, you will need to modify the CHDIR statements in each of the example programs (If you wish to obtain a full set of .bmap files, they may be found on Amicus disk #8. I am sure they are also available through any source of public domain software for the Amiga).

You may do many things with "plain, old, ordinary text" on the Amiga! Use the readily available facilities to your advantage.

```
' Listing #1
' Examples of the Eight Different Print Styles
' Using both PRINT and Text&
' Bryan D. Catley
' June 1987
'
CHDIR ":BasicDemos"                    ' Set correct directory
LIBRARY"graphics.library"              ' Set desired Library
LOCATE 1,1:PRINT"PRINTed....."         ' Print column headings
LOCATE 1,40:PRINT"Text&ed....."
FOR Style%=0 TO 7                      ' Loop for each style
  CALL SetSoftStyle(WINDOW(8),Style%,255)
  Style$=STR$(Style%)
  TxtStr$="This is display style "+Style$+"."
  LOCATE 3+(Style%*2),1:PRINT TxtStr$
  LOCATE 3+(Style%*2),40:Display TxtStr$
NEXT
SetSoftStyle WINDOW(8),0,255           ' Reset To Normal Style
LOCATE 22,32:PRINT"Click to quit"      ' Tell how to quit
WHILE MOUSE(0)=0:WEND                  ' Wait for mouse click
LIBRARY CLOSE                          ' Close the Library
CHDIR ":"                              ' Reset directory
END

' Sub-program for use of Text& Function
'
SUB Display (Txt$) STATIC
  CALL Text&(WINDOW(8),SADD(Txt$),LEN(Txt$))
END SUB
```

```
' Listing #2
' Examples of the Eight Different Display Modes
' Bryan D. Catley
' June 1987
'
SCREEN 1,320,200,3,1 ' Lo-Res, 8 Color Screen
WINDOW 2,,,0,1       ' Full Size Window, No Gadgets
PALETTE 0,.3,.3,.3   ' Gray, complement is 7 (White)
PALETTE 1,1,0,0      ' Red, complement is 6 (Not Used)
PALETTE 2,0,1,0      ' Green, complement is 5 (Not Used)
PALETTE 3,0,0,0      ' Black, complement is 4 (Yellow)
PALETTE 4,1,1,0      ' Yellow, complement is 3 (Black)
PALETTE 7,1,1,1      ' White, complement is 0 (Gray)
' N.B. A Palette's complement palette is the highest available
' palette number, less the number of the palette in question.
COLOR ,0:CLS         ' Clear Screen to Black Background
DIM Txt$(7),Pat%(7)  ' Dimension Two Arrays
Txt$(0)="Standard JAM1 mode......."
Txt$(1)="Standard JAM2 mode......."
Txt$(2)="JAM1 in COMPLEMENT......."
Txt$(3)="JAM2 in COMPLEMENT......."
Txt$(4)="JAM1 in INVERSVID........"
Txt$(5)="JAM2 in INVERSVID........"
Txt$(6)="JAM1/COMPLEMENT/INVERSVID"
Txt$(7)="JAM2/COMPLEMENT/INVERSVID"
RANDOMIZE TIMER              ' Initialize Random Numbers
FOR n=0 TO 7                 ' Create Random Pattern
  Pat%(n)=INT(RND*32000)
NEXT
PATTERN ,Pat%                ' Establish Pattern
COLOR 7,3                    ' White on Blue
CIRCLE(160,100),88           ' Draw circle
PAINT(160,100)               ' Fill with random pattern
CHDIR ":BasicDemos"          ' Set Correct Directory
LIBRARY"graphics.library"    ' Set up Correct Library
COLOR 1,2                    ' Red on Green
FOR n=0 TO 7                 ' Show All 8 Examples
   FOR x=1 TO 2500:NEXT      ' Delay Between Lines
   LOCATE 6+(n*2),2          ' Point to Print Location
   Mode&=n                   ' Get Mode& value
   CALL SetDrMd&(WINDOW(8),Mode&) 'Set Drawing Mode
   PRINT"Example of: ";Txt$(n)    'Print Sample
NEXT
WHILE MOUSE(0)=0:WEND        ' Wait for Click
LIBRARY CLOSE                ' Close Library
CHDIR ":"                    ' Reset Directory
WINDOW CLOSE 2               ' Close Window
SCREEN CLOSE 1               ' Close Screen
END
```

•AC•

TeleGames

"The ability to hook up with an opponent thousands of miles away puts TeleGames in a class of its own"

by Michael T. Cabral

Your mind aches for a test of sharpness, strategy and saavy. A game, perhaps? Chess would certainly do nicely . . . or backgammon . . . or even checkers.

You're out of luck, though. There's not an opponent to be found for miles around. Your favorite chess and backgammon foes are out doing "mindless" things. Your little brother, a tolerable checkers opponent, is glued to the tube, a slave to ALF.

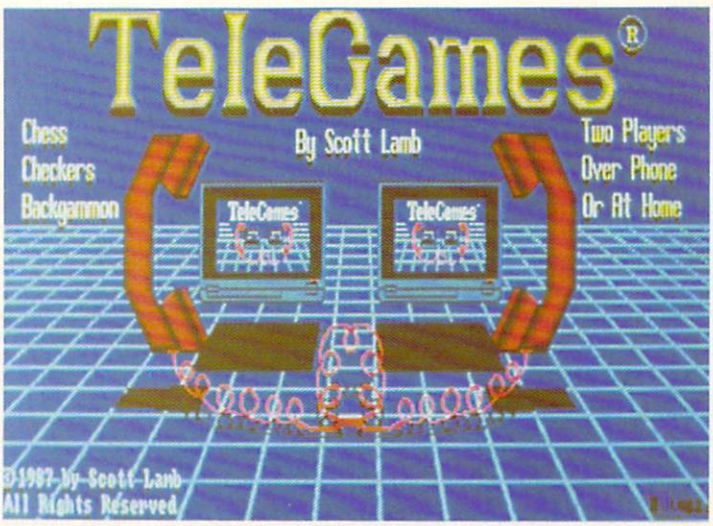
Well, just as in Spielberg's *Poltergeist*, the answer comes from the phone. TeleGames, developed by Scott Lamb and distributed by Software Terminal, allows you to wrangle with any opponent on any corner of the globe via modem!

TeleGames springs great new possibilities for Amiga gameplayers and telecommunications enthusiasts. For "telecommies" (not to be confused with Marxist, Lily Tomlin-like telephone operators), TeleGames bursts into a field which has been explored only sparingly.

For game junkies, TeleGames links you to any opponent, anywhere, anytime. You can also keep your games right at home, simply by leaving the telecommunications option deactivated and taking on a local enemy.

TeleGames . . . Starting up

TeleGames boots in normal fashion, with a Kickstart 1.2 level of ROM followed by the TeleGames disk. Pictured on the title screen are two tiny Amigas, complete with shadows, encircled by connected telephone receivers. This image is also depicted, in mini, on the screens of the shrunken



Amigas. Not bad for a title screen — a clue to the fine attention to detail and impressive graphics yet to come.

A click in the title screen prompts a "validation character" requester. A quick reference to the TeleGames manual for the proper letter frees the program. Otherwise, TeleGames is not copy-protected, so you can make as many copies as you need. The manual itself is printed in a very light type, similar to non-photographic blue, to protect against free-for-all distribution.

TeleGames . . . Up and running

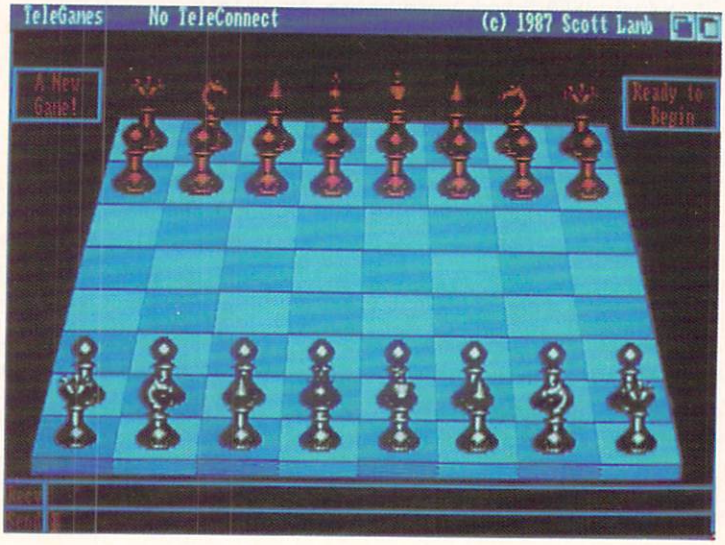
After validation, TeleGames unmasks its graphic power. The screen becomes a three-dimensional chess board, complete with 3-D playing pieces. Message boxes at upper left and upper right let you know this is "a new game" and you are "ready to begin." A click and hold on the right mouse button cues all available menus.

The first menu, titled "TeleGames," presents your game choices: TeleChess, TeleCheckers, TeleBackgammon or Exit TeleGames. Clicking any of the games places a convenient requester box at center screen that is, essentially, a safety valve. You are asked if you would like to "continue with this action," so you can escape any error you may have made. Certainly a nice, sanity-saving feature.

If you confirm your choice, you are whisked off to your selected game. Once you reach your game, the second menu allows you to start a game, end a game or choose which color pieces you will control. You will want to bypass the last two options until after you've chosen your "Board View" options from the third menu.

continued...

According to the "Board View" menu, you can view the board in either two or three dimensions. The 3-D images hold true to the title and are quite impressive. The images are crisp, well-shaded and really do give the game a realistic three-dimensional look.



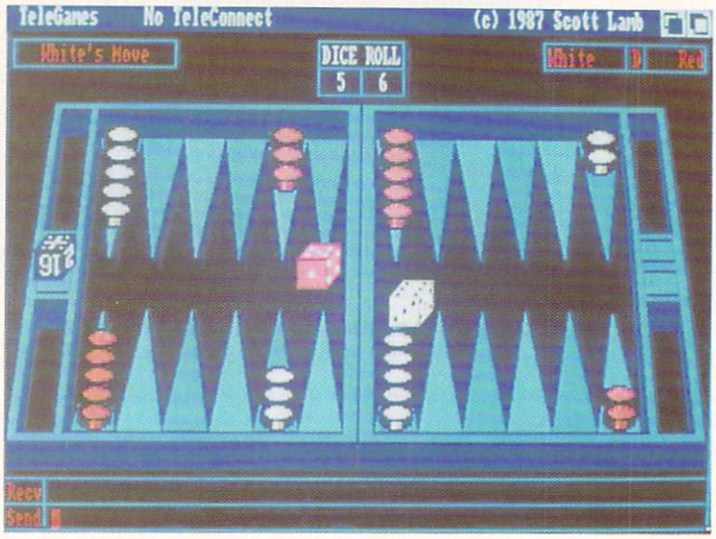
The 2-D viewpoint is, basically, an overhead shot. Again, the graphics themselves are good, but I really cannot see why anyone would refuse the 3-D angle. The game pieces, especially the chess pawns, are much smaller and further from the real thing in 2-D. The 2-D is there if you prefer, but the true strength of the graphics shines best in 3-D.

In addition to the choice of dimensions, you are also free to choose your "board view angle," your point of view in relation to the gameboard. This feature actually lets you "sit" wherever you wish. "Red's view" and White's view" position you closest to your own pieces, as if you are sitting at a table and these pieces are right in front of you.

If you are playing TeleChess, you can also opt for "King's View" or "Queen's View." You can now be seated alongside the board, so you have a lateral view of the entire layout. Such choices of viewpoint add another very nice feature to TeleGames. You and your opponent can each check out your moves from the point of view from which you feel most comfortable.

The next two menus, "Telecomm" and "TeleFiles," deal with TeleGames' primary focus — games by modem. We will come back to these later when we talk specifically about the telecommunications connection.

The "Game Files" menu lets you save a game in progress and resume at a later time. This easy-to-use feature is very helpful, especially when playing chess. As you chess vets out there know, chess can be a looonnnng and involved game. It is great to be able to halt a game mid-stream and return refreshed and ready for intense strategy. Chess, backgammon and checkers games in progress are saved separately in neatly organized individual file "folders." A click on the "Game Directory" option smoothly slides the listing over the current screen and you are free to resume any previously saved contest.



Game sounds can be switched on and off with a final menu. TeleGames audio is limited, but pleasingly unobtrusive. Digitized chimes signal important game choices and clicks mark movements of game pieces. When playing thoughtful, strategy-oriented games, too much sound can be a real irritation. TeleGames incorporates just enough sound to keep the players sufficiently informed.

TeleGames . . . Gameplay

Once you have chosen your game, dimension view and board view, you are ready for action. A click on "start new game" under the second menu sets things in motion.

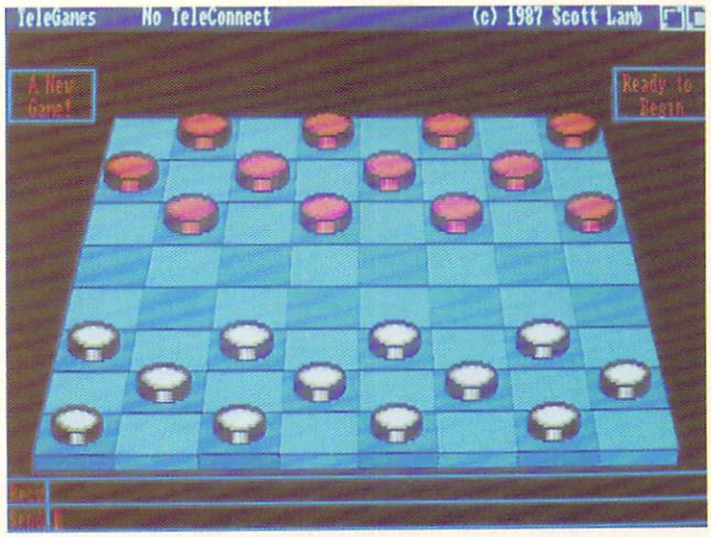
Movement of the game pieces is handled smoothly by the mouse. The keyboard comes into play only for copyright verification at the outset, and sending messages once a teleconnection has been established.

The mouse is simply a natural extension of your hand. Moving the pointer to the top of a piece, backed by a click and hold on the left button, activates the piece. That piece can then be moved wherever you wish (within the rules of the game), and will not be released until you let up on the mouse button.

It is very easy to forget you are not actually using your own hand. Control with the mouse is effortless and nearly flawless. It is nice to not have to worry that your piece may land on the wrong spot. Slick mouse control ensures that all "bonehead" plays rest squarely on your own shoulders.

Messages boxes at the upper left and upper right of all games act as electronic referees, keeping a close watch on all action. When an attempted font is disallowed, it's nice to know why. The boxes do a great job of keeping you informed.

In TeleChess and TeleCheckers, the left box keeps track of who must make the next move. The right box checks the legality of all moves. If you try to sneak a shady move by the ref, your piece will be returned to its original spot. The right box also monitors all special game situations. In checkers, for instance, if a jump must be made, the box lets you know. If you are in check in a chess match, the right box fills you in.



In TeleBackgammon, the message box arrangement is slightly different. Turns and game moves are both handled by the left box. The right box, in turn, acts as a scoreboard (cued with the left mouse button) and in gameplay, as the "doubling die" indicator. The TeleBackgammon display also includes a "dice roll" gadget.

According to the TeleGames manual, TeleChess, TeleCheckers and TeleBackgammon follow the "official rules" of the games. Although the purpose of this review was not really to pick apart the individual games, all three do appear to run true to form. I did not run across any discrepancies in gameplay or rule interpretation.

Under the realm of "official rules," TeleGames supports many options. TeleChess includes castling and la passant. Promoting pawns is handled nicely by a series of concise, self-explanatory requesters. Check and checkmate are indicated in the right message box.

TeleCheckers allows all standard "jumps," including multiples. The multiple capture procedure is somewhat different from hopping around on your kitchen table — TeleCheckers accomplishes multiple jumps as a series of single jumps. Normally, your turn would end after a single move, but TeleGames recognizes the multiple possibility, and lets you continue until no other jumps can be made. Crowning of pieces is taken care of in standard form, with typical freedom of movement.

TeleBackgammon follows all standard rules, including the "Doubling Die." At any point, the player in control of the doubling die can offer it to his opponent, simply by clicking on it. A requester makes sure you want to offer the option.

TeleGames . . . Teleconnection

The ability to hook up with an opponent thousands of miles away puts TeleGames in a class of its own. No other game can offer the versatility and opportunity for expansion explored by TeleGames. You might think such a unique feature is a hassle to hook up. Think again. The teleconnection is a breeze with TeleGames.

After both you and your opponent boot TeleGames, you must establish the ability to communicate with your own modem and your opponent's modem. This task is taken care of easily enough, under the "TeleFiles" menu, where you must first set your own modem parameters.

A click on the "Modem Parameters" option slides the necessary settings — Baud Rate, Dial Command and Answer Command — onto the screen. The positive feature here is that you need set your parameters only once — from then on, your settings will be saved and loaded automatically.

Baud rates of 300, 1200, 2400 and 9600 are supported and can be selected with a simple click.

For dial and answer commands, TeleGames adheres to the "AT" command set. Blank boxes are provided for your own dialing and answering commands or additional modem commands.

Just as you need set your own parameters only once, the "Opponent Parameters" option also saves you some trouble by allowing you to load previously saved opponent files. Entering and saving your foe's telephone number and the baud rate at which you will compete is all it takes. That opponents file will now be in memory and available for quick, easy access.

continued on page 110

SYMPHONY MUSIC VIDEO

SYMPHONY MUSIC VIDEO continuously displays pictures and plays music. At the end of a selection, another picture and music piece is loaded. The music and pictures are all in IFF format so you can modify ours with your favorite Paint or Music program or use your own music and pictures and create your own MUSIC VIDEO. You can even have your MIDI synthesizer play the music.

The MUSIC VIDEO that is included is perfect for the Christmas Season. Traditional Christmas scenes and music are continuously played. Fun to listen to and watch. Adds to the best of seasons. **$24.95**

Symphony Songs

Ready to Play Music For

Deluxe Music, Music Studio, Sonix

A library of nearly 1,000 music masterpieces ready to play with your favorite music program. All selections are in both IFF and MUSIC STUDIO format. Space does not allow listing all songs in each volume, however, a few titles, the number of songs, and the total playing time is given.
Complete list of songs **$3.95.**
Each volume is **$24.95** and includes the complete list.

BEATLES Part 1
Vol 15 (21 Pieces 40 Min)
Let It Be, Yesterday, Eleanor Rigby, When I'm 64, . . .

BEATLES Part 2
Vol 40 (15 Pieces 40 Min)
Magical Mystery Tour, Lucy In The Sky With Diamonds, Penny Lane, . . .

CLASSICAL Part 1
Vol 27 (18 Pieces 40 Min)
Prelude #1, Moonlight Sonata 1st and 2nd Movement, . . .

CLASSICAL Part 2
Vol 34 (13 Pieces 40 Min)
Sonata In C Major, Jesus Joy Of Man's Desire, . . .

CLASSICAL Part 3
Vol 31 (14 Pieces 35 Min)
1st Piano Concerto, Polonaise Sonata In C Major, Etude #3, . . .

CLASSICAL Part 4 (Bach)
Vol 35 (22 pieces 30 Min)
Two Part Invention #1, Three Part Invention #6, Prelude and Fugue 1, . . .

CLASSICAL Part 5 (Bach/Clementi)
Vol 46 (24 Pieces 50 Min)
Choral #1, Sonata #1, Theme and 11 Variations From The 2nd Sonata, . . .

BEETHOVEN, BROADWAY, & BLUES
Vol 38 (15 Pieces 40 Min)
2nd Movement Of the Pathetique Sonata, Minuet In G, Fuer Elise, . . .

COUNTRY CLASSICS Part 1
Vol 41 (15 Pieces 45 Min)
Thank God I'm a Country Boy, Act Naturally, . . .

ROCK Part 1
Vol 32 (19 Pieces 50 Min)
AXEL F, Eye Of The Tiger, Both Sides Now, . . .

ROCK Part 2
Vol 16 (20 Pieces 40 Min)
Georgy Girl, Guantanamera, Theme From "Love Story," Cherish, . . .

80's GREATEST
Vol 24 (15 Pieces 50 Min)
Hill Street Blues Theme, Chariots Of Fire Theme, Dynasty Theme, . . .

70's GREATEST
Vol 12 (19 Pieces 45 Min)
Tie A Yellow Ribbon On The Old Oak Tree, We've Only Just Begun, . . .

60's GREATEST
Vol 13 (19 Pieces 45 Min)
Windy, By The Time I Get To Phoenix, Come Saturday Morning, . . .

GOLD & PLATINUM HITS
Vol 45 (19 Pieces 60 Min)
Thriller, 99 Luft Balloons, California Girls, . . .

KENNY RODGERS HITS
Vol 39 (12 Pieces 45 Min)
Lady, Ruby, She Believes In Me, The Gambler, . . .

BILLY JOEL GREATEST HITS
Vol 43 (17 Pieces 65 Min)
Piano Man, Say Goodbye To Hollywood, Only The Good Die Young, . . .

COUNTRY CLASSICS Part 2
Vol 42 (13 Pieces 50 Min)
Ode To Billy Joe, Me and Bobby McGee, Country Roads, . . .

TV THEMES
Vol 37 (21 Pieces 35 Min)
Hill Street Blues, St. Elsewhere Theme, Masterpiece Theater Theme, . . .

MOVIE THEMES
Vol 19 (16 Pieces 40 Min)
MASH Theme, The Rose, Can You Read My Mind (Superman), . . .

BROADWAY'S THEMES
Vol 47 (25 Pieces 65 Min)
The Last Supper, Dr. Doolittle, The Old Dope Peddler, . . .

CHURCH MUSIC
Vol 23 (26 Piece 50 Min)
Amazing Grace, What A Friend We Have In Jesus, . . .

BARBERSHOP
Vol 22 (22 Pieces 45 Min)
Hello Dolly, Put On a Happy Face, Hey Look Me Over, . . .

RICHARD RODGERS SONGBOOK
Vol 18 (19 Pieces 40 Min)
Climb Every Mountain, DO-RE-MI, The Sound Of Music, . . .

NOSTALGIA
Vol 17 (22 Pieces 45 Min)
Let Me Call You Sweetheart, Ain't Misbehavin', On The Goodship Lollipop, . . .

CHRISTMAS
Vol 36 (24 pieces 50 Min)
O Little Town Of Bethlehem, Let It Snow, March Of The Toys, . . .

POLKA PARTY
Vol 33 (17 Pieces 40 Min)
Happy Polka, Pizzacato Polka, Betty Polka, . . .

We accept CASH, CHECK, COD, VISA and MASTER CARD orders.
Shipping and handling US and Canada $3.00
Shipping and handling outside the US and Canada $5.00
COD charge $2.00
Illinois residents add 6¼% sales tax.

VISA MasterCard

Dealer Inquires Invited

Speech Systems
38W255 DEERPATH ROAD
BATAVIA, ILLINOIS 60510
(312) 879-6880

Deluxe Music Construction Set is a trademark of Electronic Arts. Sonix is a trademark of Aegis. Music Studio is a trademark of Activision.

Funky SoundScape Programming

Intercepting MIDI events

Dig a bit deeper into the innards of the SoundScape system and write a module that is not displayed on the Patch Panel at all ...a **VU Meter**

by Todor Fay – author of SoundScape

Introduction

A flashy and somewhat useful module for SoundScape would be a VU meter. We could implement the VU meter as a typical module that sits on the right side of the Patch Panel and displays flashing lights whenever modules patched to it on the left send note events . . . but that would be too mundane. Instead, we'll dig a bit deeper into the innards of the SoundScape system and write a module that is not displayed on the Patch Panel at all. Rather, it works by intercepting all events sent to the Sampler, processing them and then passing them on.

This approach involves a little more knowledge of MIDI message passing, some new data structures and rules for installing intercept code. Let's start with these points and work up to the VU meter module.

If you haven't read the first article on SoundScape programming, it will help explain much of what is to come. You'll find that article in Volume 2, Number 5 of Amazing Computing.

Four more SoundScape library routines

Before we dive in, I'd like to introduce four SoundScape library calls, two of which will be used in the VU Meter module.

OpenMidiPort(port_id,direction)
Rather than allowing the user to activate a port, you can do it directly by making this call. 'Port_id' is the ID number of that port (between 1 and 255). 'Direction' is set to 1 to open on the left side of the Patch Panel, and is also cleared to open on the right side. In addition to actually calling the opencode that the port provided to SoundScape, this setting keeps track of the fact that the port is now opened.

If the port opens successfully, TRUE is returned, else FALSE. If the port is already open, only TRUE is returned.

CloseMidiPort(port_id,direction)
As the inverse of OpenMidiPort, CloseMidiPort deactivates the port in the specified direction, instead of activating it. This command calls the port's closecode, if the port was not already closed.

TRUE is returned if the port is successfully closed, FALSE if not.

EditMidiPort(port_id,direction,command,state_structure)
This call opens the port in the specified direction, then calls the port's edit routine. 'Command' and 'state_structure' deal with the nature of the edit routine. These are documented in the first Soundscape article; refer to that article for a detailed review.

OutMidiPort(port_id,note_event)
This command sends 'note_event' to the specified port. Normally, you use the Send command to send events. However, the Send command passes copies of the event to all receivers set by the Patch Panel to receive from your port. You may wish to send an event to one specific receiver. Use this command to do so.

The Port Structure

When a module first installs itself as a port in the Patch Panel, it does so by calling AddMidiPort(), passing two icons to it to display a name and ID number, and passing pointers to four routines. As it turns out, nearly all this information is stored in one data structure, the Port structure. You can access the Port structure of every module installed in SoundScape.

First, we have a small structure used to keep track of all ports a particular port is sending to. These ports are maintained as a linked list. Each node in this list has the standard SoundScape link structure and the id of the port to send to:

continued...

```
struct Ports {
   struct Link link;     /* Link information. */
   unsigned char port;   /* Port to send to. */
};
```

This linked list of port ids is graphically represented in the Patch Panel (as all the lines connecting from a port on the left to the ports on the right being sent to). It is a linked list because the number of ports it sends to can fall between none and all of the ports.

Then, we have the Port structure:

```
struct Port {
   struct Ports *portsout;  /* Ports to send to. */
   char *name;              /* Each has a name. */
   long (*outcode)();       /* Output routine. */
   long (*editcode)();      /* Edit routine. */
   long (*closecode)();     /* Close routine. */
   long (*opencode)();      /* Open routine. */
   char countin;            /* How many sending here. */
   char doin;               /* Sending active flag. */
   char doout;              /* Receiving active flag. */
   unsigned char show;      /* ID of thief. */
};
```

SoundScape maintains an array of 255 pointers to Port structures. Initially, all these slots are empty.

When a port is created by the AddMidiPort command, a Port structure is created and stuffed with the four routines and the name. This structure is then placed in the array, at the index corresponding with the port ID.

An array is used because it provides for very fast accessing – it provides a table look-up to get at the Port structure, rather than searching a linked list. This speed is very important because the MIDI message passing operations must be fast.

The 'portsout' field is a pointer to the linked list of Ports structures defining the port to be sent to.

The 'name' field points to the string given by the AddMidiPort call. This process takes place so all ports can be identified by names, as well as IDs. This arrangement is useful because ports need not be bound to particular id numbers.

We then have four pointers to the four routines that a port must provide. A brief summary of these routines: 'Outcode' is a routine called to process a MIDI note event sent to this particular port. 'Editcode' provides the user interface and code to handle data sharing with other modules. 'Closecode' is a routine called when another module, or the system itself, wishes to deactivate the chosen port. 'Opencode' is called to activate this port.

Since you can access the Port structure for any port, accordingly, you can call any of the four routines yourself, rather than going through one of the four SoundScape library calls outlined above. However, those calls do a fair amount of extra work for you. Unless you have some strange and compelling reason, it's best to do these calls in the proper manner.

'Countin' simply keeps track of how many ports are sending to this specific port. If a port has a countin of three, there must be three ports on the left with patchcords connecting to this port on the right.

'Doin' is a flag set only if the selected port is activated for sending MIDI events. This flag indicates that the open routine was called, requesting to open this port on the left hand side of the Patch Panel . . . and it was successful.

'Doout' is a flag that is set only if the port is activated for receiving MIDI events. If set, the open routine was called to open this port on the right side of the Patch Panel . . . and it also was successful.

The terminology of 'input' and 'output' can be confusing at times. It may seem odd to refer to something that sends events as input, while something that receives them as output. From the point of view of the message passing system, though, ports that send events are, indeed, inputs, while those that receive are outputs.

How a Message is Sent

We've talked previously about how to allocate and send MIDI events. Now, let's see what goes on once the Send command is issued.

Send(port_Id,note_event);
Send stamps 'note_event' with 'port_id', the port ID of the sender. 'Note_event' is then placed in a queue of events waiting to be sent.

This queue is processed by a task running at a priority of 15. This task is called the event router. When the queue is empty, the event router waits. When an event is placed in the queue, it is alerted with an Exec Signal() command.

The event router reads the id of the sender from each event. It can instantly get the Port structure from the array of Ports with a quick table look-up. From the Port structure, the event router gets the linked list of ports to be sent to. For each of these ports, once again, a port ID is used to access

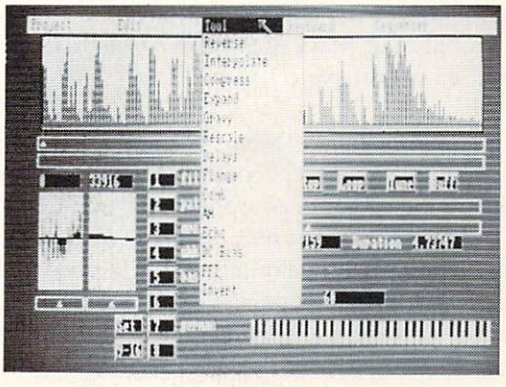
continued on page 78

Rediscover Sound

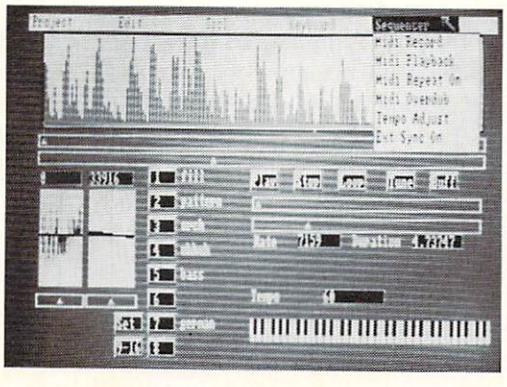
Sit Down at your Amiga with Studio Magic and discover a new way—a better way—to create sound...

Because you'll be working with software modeled after professional digital sound studios.

The Studio Magic tool menu is packed with over a dozen different digital tools. These special effects let you do anything from creating echos, flanges, and delays to enhancing frequency components or performing Fast Fourier Transforms. The Studio Magic tool box puts the power of a digital sound studio at your finger tips. Imagine the possibilities! You can take a voice and make it sound old or young, like Darth Vader, or like an alien from mars. Effects like M-M-Max Headroom become childs play. Your video sound tracks will never be the same!



Studio Magic includes over a dozen digital effects including echos, flanges and delays.



Studio Magic's midi support includes a sequencer with overdub and external sync.

But Studio Magic doesn't only do sound effects... it also does midi. You can record in real time. You can overdub. You can assign any sound to any key on your midi keyboard. Studio Magic even supports advanced midi features such as tempo adjust and external sync. When you combine Studio Magic's midi support with its tools, editing capabilities, and keyboard control, you have the *power* to create spectacular musical pieces. Your amiga never sounded so good.

Studio Magic is IFF compatible. You can import any IFF digital sound, and save in IFF instrument or one-shot format. And of course, Studio Magic works with the Perfect Sound audio digitizer.

Run, don't walk, to your dealer for a demo today!
Suggested retail price **$99.95.**

Studio Magic
SunRize Industries

SunRize Industries
3801 Old College Road
Bryan, Texas 77801
(409) 846-1311

continued from page 76

the Port structure of each port to be sent to. To send an event, the event router makes a copy of the event it is holding, and then makes a subroutine call to the 'outcode' routine provided in the destination's Port structure. The copied event is passed as a parameter.

The MIDI event is now in the hands of the receiver. The receiver does what ever it wants with the event, then returns control back to the event router, which, in turn, continues sending the remaining events.

That's SoundScape's MIDI message passing mechanism.

Intercepting MIDI events

To intercept the events being sent to a particular port, all a module must do is install its own output routine in the Port structure of the port and maintain the previous output routine for itself. The event router passes the event directly to the routine, and it can do what it wants before passing the event on to that port.

There is one potential problem. It is possible for several modules to install intercept routines in the same Port. Each installs its code and we end up with the event being passed down a chain of routines, each doing its own thing. All is fine and dandy, until one of these tries to close down. It will incorrectly place the pointer to the following routine back in the Port structure, leapfrogging all those routines that precede it. Worse, after that, when the routine that preceded it removes itself, it will reinstall the latter's routine back in the Port structure. The latter's segments may have already been removed by AmigaDos, so when the next event comes along . . . GuruLand.

There must some way of regulating this action. There is one last field in the Port structure, called 'show," that I haven't mentioned. Normally, it is cleared, but if some other module has inserted itself in this port, the port id for that module's port is stored here. This storage lets other people know they shouldn't mess with this port — somebody else already has. However, since they know which port has it, they can coordinate with the field . . . or they can ask that port to leave by closing it with a call to CloseMidiPort.

By the way, don't ask where the name 'show' comes from.

Here are the steps for creating your intercept routine:

Provide a variable to store the address of the port's output routine.

Write your intercept routine. It should call the port's original output routine when finished.

Create your own port. The four routines for it should be:

OutCode – does nothing.

EditCode – does nothing.

OpenCode – Installs your intercept routine in the intended port. If someone else has this port (look at port->show), follow one of two options: either don't do anything and return(0) for unsuccessful open, or call CloseMidiPort(port->show), which instructs that other port to deactivate and remove itself from the Port. You can then safely install your intercept routine.

CloseCode – Removes your intercept routine.

Here's an example intercept output code that does nothing; it just passes the event on to the port. The port's outcode routine is pointed to by the variable 'portoutcode.' Note that this variable is used to pass the event on, rather than using the OutMidiPort command. That command would simply route the event back to this routine, since the Port structure points here.

```
void (*portoutcode) ();
interceptcode(note_event)
struct Note *note_event;
{
   (*portoutcode) (note_event);
}
```

Installing this intercept code is simple. Access the Port structure for the port by calling MidiPort (FindMidiPort (portname)). FindMidiPort, given a port's name, returns the ID of the port. MidiPort, given a port's ID, returns a pointer to its Port structure. If there is no such port, NULL is returned. Check the 'show' field of the Port structure. If it is non-zero, this response means some other port has already monkeyed with this field. If so, we to close it down with a CloseMidiPort command. If successful, we proceed.

```
opencode(direction)
char direction;

{
   struct Port *port;
   if (!direction) {
      port = MidiPort(FindMidiPort("port name"));
      if (port) {
         if (port->show) {
            CloseMidiPort(port->show,0);
         }
         if (port->show) return(0);
         portoutcode = port->outcode;
         port->outcode = interceptcode;
         port->show = thisport;
         return(1);
      }
   }
   return(0);
}
```

Closing the port is simple. Just reshuffle the pointers, so the port has its own outcode back in place. Clear port->show, so other modules can get at this port.

```
closecode(direction)

char direction;

{
   struct Port *port;
   if (!direction) {
      port = MidiPort(FindMidiPort("port name"));
      if (port) {
         port->outcode = portoutcode;
         port->show = 0;
         return(1);
      }
   }
   return(0);
}
```

Now that we have a method for intercepting MIDI events down pat, let's write that module.

A VU Meter

We'll read the MIDI velocity of notes going to the Sampler and use that value to calculate an overall volume. When notes are turned off, their initial velocity is subtracted from the volume. The total volume is displayed as a row of lit LEDs.

Of course, just reading the note velocity can be misleading, since some sounds may be quieter than others, and we are ignoring their amplitude envelopes. As you'll see, though, when you get the thing running, the effect is quite acceptable.

To keep track of the total volume, we can't simply keep one variable around and add velocities to it when notes are turned on. We also need to subtract the original note on velocity from the total when the note is turned off . . . but the note off event doesn't carry that information. So, we must keep the original velocity on hand. We use an array of note on velocities to accomplish this task. Whenever a note is turned on, its velocity is stored in this array, indexed by the note value. When the note turns off, we index by the note value and retrieve the velocity, which we subtract from the total.

The VU meter is displayed as a line of simulated red LEDs. The number of LEDs to light up is calculated from the total volume. It is not a simple linear relationship. Rather, it makes more sense to illuminate one more LED for every doubling of volume. This figure corresponds to a six dB increase. The number of LEDs to be displayed is calculated by finding what power of two the total volume equals. This calculation can be done with a series of shift instructions, since a shift left, in binary doubles the value.

Displaying the LEDs is very simple. There are two Intuition Images; one is a blank rectangle, indicating an off LED, and the other is a red rectangle, indicating an on LED. Instead of redrawing the entire display for every change in volume, only the LEDs that constitute the change are updated. For example, when the volume decreases by two LEDs, the top two LEDs are cleared by drawing with the blank image.

Installing the VU Meter

Since this SoundScape module does not have an icon in the Patch Panel, there is no way for the user to invoke it by simply clicking on that icon.

There are two solutions to this problem.

You can put an icon in the Patch Panel that does nothing other than provide a means for opening the VU Meter.

You may also have the VU Meter open automatically when it is loaded as a module. Immediately after it installs itself in SoundScape with an AddMidiPort call, it opens itself with a call to OpenMidiPort.

I chose the latter solution to avoid cluttering the Patch Panel with the extra icon.

(see listing for vu.c)

Here's the .with file to link this feature:

```
FROM *
Lib:Astartup.obj,*
vu.o,*
sslink.o,*
morelib.o
TO *
VU
LIBRARY *
Lib:amiga.lib,*
Lib:lc.lib
NODEBUG
```

I use Blink. If you have Alink, remove the NODEBUG statement.

Compile and link this program. Get SoundScape running and run VU from the CLI (or click on it, if you've made a WorkBench icon.). The VU Meter should pop up in the top left hand corner. Connect the Console Keyboard to the Sampler. Play notes. Although no sounds may be loaded into the Sampler yet, the LEDs will flash . . . filthy liars.

Run VU a second time. The VU Meter window should close down, then reopen. This occurrence comes up because the second VU Meter closed down the first, in order to steal the Sampler.

continued...

Intercepting MIDI events has many uses. Here are a few ideas:

• A module that intercepts CLOCK events going to the Tape Deck and displays the time in beats and measures.

• A diagnostic module for debugging that filters events going to a user specified port.

• An insidious module that sometimes adds strange transpositions to events being sent to the Sampler or out the MIDI connector. It could also randomly reroute events destined for the Sampler to the Player Piano and Mixer.

• This program and all necessary link files will are available on an Amicus disk, as well as on PeopleLink. There are versions for both Lattice and Aztec C.

```
/*      VU.C            VU Meter for the Sampler
        (c) 1987        Todor Fay

LATTICE C version:

lc1 vu
lc2 -v vu
atom vu.o ram:foo -cd
copy ram:foo vu.o
blink with vu.with
*/

#include "exec/exec.h"
#include "exec/types.h"
#include "soundscape.h"
#include "intuition/intuition.h"

/*      Images for the LEDs.  A red LED for lighting,
        a blue LED for turning it back off
        (the background color.)
*/

UWORD redleddata[] = {          /* 16 x 3 */
65535,
65535,
65535,
65535,
65535,
65535,
};

struct Image redledimage = { 0,0,16,3,2,redleddata,3,0,0 };

UWORD blueleddata[] = {         /* 16 x 3 */
0,
0,
0,
0,
0,
0,
};

struct Image blueledimage = { 0,0,16,3,2,blueleddata,3,0,0 };

/*      We don't need any Intuition Gadgets, just a
        window to display the LEDs in.
        This was designed with Power Windows.
*/

struct NewWindow NewWindowStructure = {
        0,0,
        250,20,
        0,1,
        NULL,
        WINDOWDRAG+WINDOWDEPTH,
        NULL,
        NULL,
        "VU Meter",
        NULL,
        NULL,
        5,5,
        0,0,
        WBENCHSCREEN
};

short thisport;                         /* This port's id. */
long SoundScapeBase;                    /* SoundScape library. */
long  IntuitionBase;

/*      We need a place to store the Sampler's
        output routine address, since we are replacing
        the code pointer in its port structure
        with our own.
*/

void (*sampleroutcode) ();
struct Window *vuwindow;                /* Display window. */

/*      We need an array of velocity values for each
        note that is being played.  These are zeroed
        for no note.
        If you add up all the velocities in this
        array, you should get totalvolume, which
        is used to determine the number of LEDs to
        light.
*/

unsigned char velocity[128];
unsigned short totalvolume;

insertoutcode(note)
```

```
/*      This output routine is not given to the Patch
        Panel in the AddMidiPort call.  Instead, this
        routine replaces the Sampler output code, and
        then calls it.  So, this is inserted there by
        the open routine.

        Check the note status.  If it is NOTEON or NOTEOFF,
        we want to use it for the display.

        Start by decrementing the totalvolume by
        velocity[note->value].  If this is a NOTEOFF
        event, this reduces totalvolume appropriately.
        If this is a NOTEON event, this clears any
        previous NOTEON that was stored.  This is
        neccessary because we aren't supporting multiple
        playing of the same note (for simplicity.)

        If this is a NOTEON event, increment totalvolume
        by the velocity of this note.  Store this
        velocity in the velocity array.

        Else, for a NOTEOFF event, simply put zero in
        the velocity array - there is no note being
        played.

        Compute how many LEDs should be on from
        totalvolume.  Each additional LED represents
        a doubling in volume, so figure what the most
        significant bit of totalvolume is.

        If the new top LED position (newvupos) is
        greater than the currently displayed one
        (vupos), draw in the extra LEDs.  Else,
        if newvupos is less than vupos, erase
        LEDs down to newvupos.

        Finally, forward the note event to the
        Sampler by calling its output routine.
*/

struct Note *note;

{
    unsigned char status = note->status & 0xF0;
    static short vupos = 0;
    short newvupos, index;
    if (sampleroutcode) {
        if ((status == NOTEON) || (status == NOTEOFF)) {
            totalvolume -= velocity[note->value];
            if ((status == NOTEON) && note->velocity) {
                totalvolume += note->velocity;
                velocity[note->value] = note->velocity;
            }
            else velocity[note->value] = 0;
            newvupos = 0;
            index = 4;
            for (;index < totalvolume; index = index << 1)
                newvupos++;
            if (newvupos < vupos) {
                for (;vupos > newvupos; vupos--)
                    DrawImage(vuwindow->RPort,
                        &blueledimage,-10 + vupos * 20,13);
            }
            else if (newvupos > vupos) {
                vupos++;
                for (;vupos <= newvupos; vupos++)
                    DrawImage(vuwindow->RPort,
                        &redledimage,-10 + vupos * 20,13);
            }
            vupos = newvupos;
        }
        (*sampleroutcode) (note);
    }
    else FreeNode(note);
}

outcode(note)

/*      This, the outcode for this module, does
        nothing.
*/

struct Note *note;

{
    FreeNode(note);
}


opencode(direction)

/*      This installs the VU meter.

        If this is to install the output routine:

        Get the Port structure for the Sampler.
        If the Sampler already has its output
        code stolen, (samplerport->show has somebody's
        port id) do nothing.  Else, open the window,
        give samplerport->outcode insertoutcode
        and put the Sampler's output code
        in samplercode.  Set samplerport->show to
        this port's id, so no other module will
        monkey with it.
*/

char direction;

{
    short i;
    struct Port *samplerport;
    if (!direction) {
        samplerport = (struct Port *)
            MidiPort(FindMidiPort("sampler"));
        if (samplerport) {
            if (samplerport->show) {
                CloseMidiPort(samplerport->show,0);
            }
```

continued...

```
            if (!samplerport->show) {
                vuwindow = (struct Window *)
                    OpenWindow(&NewWindowStructure);
                if (vuwindow) {
                    totalvolume = 0;
                    for (i=0; i < 128; i++)
                        velocity[i] = 0;
                    sampleroutcode = samplerport->outcode;
                    samplerport->outcode = insertoutcode;
                    samplerport->show = thisport;
                    return(1);
                }
            }
        }
    }
    return(0);
}

closecode(direction)

/*      This deactivates the VU Meter.

        Simply return the Sampler's output routine
        to its Port structure and clear
        samplerport->show.  Close the display
        window.
*/

{
    struct Port *samplerport;
    if (!direction) {
        samplerport = (struct Port *)
            MidiPort(FindMidiPort("sampler"));
        if (samplerport && sampleroutcode) {
            samplerport->outcode = sampleroutcode;
            samplerport->show = 0;
        }
        if (vuwindow) CloseWindow(vuwindow);
        vuwindow = 0;
    }
    return(1);
}

/*      The main program is standard with one addition:
        After installing this port, call OpenMidiPort
        to activate it.  This is needed because there
        is no icon for the user to click on.  We
        could have added that icon, but it would be
        one more item on the Patch Panel, cluttering
        it and causing some confusion since nothing
        can be sent directly to it.
        There is a chance OpenMidiPort will fail.  This
        is because the Sampler's output routine was
        already stolen.  If so, there is no need to
        stick around, so call RemoveMidiPort and leave.
        Normally, RemoveMidiPort is done by SoundScape
        when it closes down, but this time we'll
        do it ourselves to get out early.
*/

main() {
    IntuitionBase = OpenLibrary("intuition.library",0);
    SoundScapeBase = OpenLibrary("soundscape.library",0);
    if (SoundScapeBase) {
        CloseLibrary(SoundScapeBase);
        thisport = AddMidiPort(opencode,closecode,0,
            outcode,0,0,-1,"vu meter");
        if (OpenMidiPort(thisport,0)) {
            SetTaskPri(FindTask(0),-20);
            while (MidiPort(thisport)) Delay(100);
        }
        else RemoveMidiPort(thisport);
    }
    CloseLibrary(IntuitionBase);
}
```

•AC•

Fun with Amiga Numbers

Uncover the secrets of floating point arithmetic buried within the Amiga.

by Allen Barnett

The Amiga's speed in dealing with floating point numbers intrigues me. After a few tests with empty loops and multiplies in BASIC and C, though, I began to realize my curiosity would never be satisfied with anything less than full blown assembly language code. This article discusses my attempts to uncover the secrets of floating point arithmetic buried within the Amiga.

AMIGA NUMBERS

Before looking into the math, let's consider how floating point numbers are stored in the Amiga. Unfortunately, there are two different formats available. While the differences between formats are small, they are sufficient to render incompatibility. The two formats are: the Motorola Fast Floating Point (FFP) and the Institute of Electrical and Electronic Engineers (IEEE) standard. A good explanation of the FFP form is given in Part 4, Chapter 1 of the ROM Kernel Manual. The IEEE standard single precision format is discussed in Appendix D of the AmigaBASIC manual.

The major difference between the representations is that an FFP's exponent is stored in the least significant bits, while an IEEE number stores the exponent in the most significant bits. A more subtle difference is that FFP exponents are stored in excess-64 notation, while IEEE exponents go with excess-63. Again, I can offer no explanation. The last difference is the most interesting and merits some discussion.

In both FFP and IEEE formats, the mantissa is stored as a normalized binary fraction. This storage means the leading bit is always one. The IEEE format takes advantage of this situation by using this bit to store something else, namely an extra bit of exponent. So, while the FFP can store only 7 bits of exponent, an IEEE manages 8 bits. This difference, of course, gives IEEE numbers greater range in magnitude by a factor of two. Thus, while an FFP may range from approximately 1E-20 to 1E+18, an IEEE number can go from about 1E-38 to 1E+38 (note that the greater symmetry of the IEEE format is due to excess-63 notation). The accuracy of the mantissa of each representation is still the same - - both use 24 bits - - only the magnitude is different.

The Amiga provides basic math support (addition, subtraction, multiplication and division) for both of our formats. It also allows comparison, testing (comparison to zero), negation, absolute value and conversion from and to integers (floating and fixing). The Amiga also provides a disk-based library of transcendental functions, such as sine, cosine, square root and a variety of logarithms. These functions, however, only work on FFP numbers. Fortunately, this library also offers IEEE format conversion, so we are free to choose the form. I recommend using the IEEE format whenever possible, since it offers a greater range of representation (Avogadro's number is 6.023E+23) and it is a standard. Further, your numbers will be compatible with BASIC and C, at least on the Amiga.

Code conversion

Having filled in the basics, I now report that the program I finally wrote uses the FFP format. I used FFP for two reasons. First, the point of the exercise was to 'make the boat go fast.' Second, we are provided with routines which convert ASCII strings directly to FFPs and back again. To access these routines, though, we must delve into the arcane innards of the machine (You might want to read the first paragraph of Part 4, Section 1.5 of the ROM Kernal Manual. Then, you see what we are up against.).

No interface is provided for assembly programs to the requisite routines. Further, I could find no trace of a mathlink_lib.lib. It turns out, though, that this library is actually part of the link library, amiga.lib, included on both the assembler and the C compiler disk. I probably would have stopped at this point were it not for Gerry Hull's article in Volume 1, Number 7 of Amazing Computing on linking assembly language to C programs. Armed with his knowledge, I then set out to defeat the 'C interface.'

Of course, I failed miserably. I could not get my assembly code to work. Even C programs calling the conversion routines GURUed. Finally, 68000 manual in hand, I proceeded to disassemble the conversion routines in amiga.lib and, eventually, an error revealed itself. At least I think it is an error — I am not certain. Here is my discovery.

continued...

The routine that failed was _fpa, the conversion from a floating point number to an ASCII string. You may recall from Hull's article that arguments for C subroutine calls are pushed onto the stack before the routine is called. The subroutine then LINKs its own variable space below the bottom of the stack. The arguments are available at positive offsets from the top of this space.

The routine _fpa takes two arguments - - the FFP to be converted and the address of a place to store the resulting string. Nominally, this should amount to two long words. The 'C interface' to the actual conversion routine ended up going back one long word too far to retrieve the address of the string buffer. Apparently, it expected the FFP to be two long words long. Yet again, I must admit bafflement. At any rate, if you push a long word between the address and the FFP, it all works fine. The other routine I used to convert a string to an FFP, _afp, works without problems.

The specific methods for accessing these routines are shown below. Consider this code the 'assembly language interface.' To convert an ASCII string to an FFP:

```
move.l    #NumString,-(SP)
jsr       _afp
lea       4(SP),SP
move.l    d0,FFPSave
```

To convert an FFP to a string:

```
move.l    #StringBuffer,-(SP)
move.l    DUMMY,-(SP)
move.l    FFP,-(SP)
jsr       _fpa
lea       12(SP),SP
```

StringBuffer will now contain a string 14 characters long of the form:

1.mmmmmmmmE1ee

Amiga math

Part 4, Section 1.3 of the RKM explains how to access the floating point routines in the mathffp.library. Like dos.library or graphics.library, this library must be opened before it can be used. The OpenLibrary routine in the exec library accomplishes the opening. A typical code fragment is shown below:

```
move.l    _AbsExecBase,a6
move.l    #Math_Name,a1
jsr       _LVOOpenLibrary(a6)
move.l    d0,_MathBase
```

The OpenLibrary routine returns the memory location of the jump table for the math library in register d0. A typical call to an FFP routine is done as shown below:

```
move.l    _MathBase,a6
move.l    FFP1,d0
move.l    FFP2,d1
jsr       _LVOSPAdd(a6)
move.l    d0,Sum
```

The above code adds the FFP number in d1 to the FFP number in d0. In general, the mathffp routines also set flags indicative of the results. For example, the zero flag is set if the result in d0 is zero. This sequence makes using floating point numbers very easy.

Bear a few warnings in mind. First, division by zero is not handled very gracefully by SPDiv. In fact, it results in a GURU visit (I suppose the division routine must use the 68000 DIVU instruction. If this detects a division by zero, it generates an interrupt which sends us to the Exception Vector Table, and from there to the old Dead End Alert.). To avoid this end, check the divisor before you call SPDiv.

This process leads us to another point of confusion. Division is not a commutative operator - - 1/2 does not equal 2/1. The RKM indicates that the result of SPDiv divides d1

by d0 when, in fact, the routine does just the opposite. It also confuses the subtraction routine, SPSub. The result of SPSub subtracts d1 from d0. Finally, the arguments of SPMul are taken from d0 and d1, NOT d1 and d2, as stated (This point is correct in App. A.).

Having found all this useful information, I wanted to write a program that would interact with the user. Accordingly, I had to find out about machine language I/O. The routines for application I/O are all detailed in the AmigaDOS Developer's Manual, but I must confess that I couldn't make anything of them. At yet another dead end, I stumbled across the Amiga Programmer's Guide, by COMPUTE! Publications, Inc. This extremely useful work contains a chapter on Machine Language by Tim Victor, describing, among many things, how to do console I/O from assembler. Just what I needed.

Pulling all these ideas together resulted in Solve, the program listed at the end of this article. It is designed to show how to do some useful things with floating point numbers. The program is also useful in its own right, as a very elementary solver of simultaneous equations (no partial pivoting here), using Gauss-Jordon reduction. Except for division by zero, there is no error checking, so, be careful.

Here are the commands needed to compile and link the code, assuming you have the assembler in the external drive and have entered Solve.asm in the ram disk:

```
cd ram:
df1:c/assem Solve.asm -o Solve.o
df1:c/alink Solve.o lib df1:lib/amiga.lib to Solve
```

Oddly, the version of amiga.lib accompanying my C compiler is different from the version on the Assembler disk. My C version (3.02) gives an unresolved reference to _MathBase, unless this variable is declared and visible to the linker. You can declare _MathBase with XDEF in the code, but neither _afp or _fpa uses it. Perhaps some of the other conversion routines discussed in Part 4, Sec. 1.5 use _MathBase. The newer (?) version of amiga.lib does not even require it. You can run the program simply by typing: Solve.

When the program runs, it prompts for the rank (the number of unknowns) of your matrix. Enter a number between 2 and 9, inclusive, without leading blanks. The same holds true for input of the coefficients and solution vector values - - don't use leading blanks. The routine _afp is somewhat forgiving concerning the form of your entry. The following are all valid numbers:

```
1
1.
1.E1
1E-1
3.141592E+17
0.0046372
```

Note that E, indicating the exponent, must be capitalized and only one number may be entered per line.

Conclusions

I coded this same algorithm in BASIC just to see how it fared. A perceptible delay between the entry of the last value and the printing of the solution for a 9 x 9 matrix occurred. The machine code version has no such delay. I found this point very pleasing. Perhaps quantitative timings would be more scientific, but instant solutions are fast enough for me.

continued...

Listing One
Solve.asm

```
*
*       Program to solve a linear system using FFP numbers
*       Copyright ) 1986 by Allen Barnett
*                                Troy, NY
*
*
*       Declare externals so the linker can find them
*
            XREF      _AbsExecBase      ;base address for exec lib
                                        ;jump table
            XREF      _LVOOpenLibrary   ;open a library (exec)
            XREF      _LVOCloseLibrary  ;close an open lib (exec)

            XREF      _LVOInput         ;get input file handle (dos)
            XREF      _LVOOutput        ;output file handle (dos)
            XREF      _LVORead          ;input from open file (dos)
            XREF      _LVOWrite         ;print to an open file (dos)

            XREF      _LVOSPDiv         ;divide d0 by d1 (math)
            XREF      _LVOSPMul         ;multiply d0 by d1 (math)
            XREF      _LVOSPSub         ;subtract d1 from d0 (math)

            XREF      _afp              ;convert ASCII string to FFP
            XREF      _fpa              ;go the other way
*
*       Define macros for console I/O
*
WriteCon     MACRO                      ;File Handle, Output String
                                        ;String Length
             move.l   \1,d1             ;File Handle here
             move.l   \2,d2             ;String Address here
             move.l   \3,d3             ;Length of the String here
              jsr      _LVOWrite(a6)    ;Assumes _DOSBase in a6!!
            ENDM

ReadCon      MACRO                      ;File Handle, Input Buffer
                                        ;Buffer Length
             move.l   \1,d1             ;File Handle
             move.l   \2,d2             ;Buffer Address
             move.l   \3,d3             ;Buffer Length
              jsr      _LVORead(a6)     ;_DOSBase better be in a6!
            ENDM
*
*       Open the Dos library
*
Solve        move.l    _AbsExecBase,a6  ;OpenLib in exec library
             move.l    #DOS_Name,a1     ;load ptr to dos lib name
             clr.l     d0               ;will accept any version
             jsr     _LVOOpenLibrary(a6) ;open the library
             move.l    d0,_DOSBase      ;base address returned in d0
              beq       NoDOS           ;no return add., skip this
*
*       Open the Math library
*
             move.l    #Mathffp_Name,a1 ;load ptr to math lib name
             clr.l     d0
              jsr       _LVOOpenLibrary(a6)
             move.l    d0,_MathBase
               beq       NoMath         ;again, not there, can't run this
*
* Get the input file handle (The console is the default device)
*
             move.l    _DOSBase,a6      ;Input function in dos lib
              jsr       _LVOInput(a6)   ;the console is already open
                                        ;when the loader invokes a program
             move.l    d0,InHand        ;save the handle
*
* Get output file handle (Console also the default output device)
*
              jsr       _LVOOutput(a6)  ;again, console is open upon
loading
             move.l    d0,OutHand       ;save this handle, too
*
*       Prompt the user to enter the rank of matrix
*
                WriteCon OutHand,#Msg1,#Msg1_Len
*
*       Read in the rank of the matrix
*
             ReadCon  InHand,#RBuf,#RBuf_Len
              clr.l     d0              ;convert input num to int
              move.b    RBuf,d0         ;get first byte of buffer
              andi.b    #$0F,d0         ;strip off ASCII part
              move.l    d0,Rank         ;save the result in memory
              move.l    d0,d4           ;and in d4
              subq.l    #1,d4           ;fix # entries so DBRA works
              move.l    d4,Rank_1       ;and save it
              lsl.l     #2,d0           ;multiply rank x 4:gives the
              move.l    d0,RowWidth     ;# bytes in ea.row, save it
*
*       Prompt to enter coefficients
*
                WriteCon OutHand,#Msg2,#Msg2_Len
*
*       Read each coefficient
*
              move.l    #Matrix,a2      ;load ptr to matrix storage
              move.l    Rank_1,d4
RdCoef1        move.l     Rank_1,d5     ;read rank x rank #of coef
RdCoef2        move.l     Rank,d0       ;recover rank value
              move.l    #PBuf,a3        ;recover print buffer start
              sub.b     d4,d0           ;calculate row value
              ori.b     #'0',d0         ;convert to character
              move.b    d0,(a3)+        ;put in print buffer
              move.b    #',',(a3)+      ;a comma to separate indices
              move.l    Rank,d0         ;same idea
              sub.b     d5,d0           ;calculate column value
              ori.b     #'0',d0         ;convert to a character
              move.b    d0,(a3)+        ;put in the print buffer
              move.b    #':',(a3)       ;make it look neat
              WriteCon OutHand,#PBuf,#4 ;print the prompt

             ReadCon  InHand,#RBuf,#RBuf_Len
                                        ;read in the coefficient

             move.l    #RBuf,-(SP)      ;push input add. to stack
              jsr       _afp            ;call C fnc to convert the
                                        ;string into an FFP
              lea       4(SP),SP        ;clean up the stack
             move.l    d0,(a2)+         ;save FFP into matrix

              dbra      d5,RdCoef2      ;get the next coefficient
              dbra      d4,RdCoef1      ;cont until matrix is full
             move.l    a2,MatTop        ;save the end of the matrix
*
*       Read in the values in the solution vector
*
              WriteCon OutHand,#Msg3,#Msg3_Len ;print prompt for sol.
vec.
             move.l    #Solution,a2     ;load ptr to the solution
                                        ;vector storage area
             move.l    Rank_1,d4        ;get this many values
RdSol          move.l     Rank,d0       ;recover rank value
              sub.b     d4,d0           ;calculate position
```

```
            ori.b    #'0',d0            ;make it a character
            move.b   d0,PBuf            ;copy char to print buffer
            move.b   #':',PBuf+1        ;neaten the output
            WriteCon OutHand,#PBuf,#2   ;print the value

            ReadCon  InHand,#RBuf,#RBuf_Len  ;read solution value

            move.l   #RBuf,-(SP)        ;push string add. onto stack
            jsr      _afp               ;call the C function
            lea      4(SP),SP           ;clean up the stack
            move.l   d0,(a2)+           ;save FFP in storage area

            dbra     d4,RdSol           ;cont until all sol's read
            move.l   a2,SolTop          ;save ptr to sol vec. top
*
*       Solve the simultaneous system
*
            move.l   _MathBase,a6       ;gonna do math for a while
            move.l   MatTop,a2          ;recover ptr to matrix end
            move.l   SolTop,a1          ;recover ptr to solution end

            move.l   Rank_1,d3          ;do for each row i
BigLoop      move.l    Rank,d0          ;the next few instct calc
            sub.b    d3,d0              ;offset from END of a row
            lsl.l    #2,d0              ;to the diagonal element
            neg.l    d0                 ;=-(4*(rank - row_counter))
            move.l   d0,d2              ;save this factor for later
            move.l   0(a2,d0),d1        ;diagonal element of row i
            beq      DivBy0             ;if zero, that's all we do

            move.l   Rank_1,d4          ;for each element of row i
            move.l   a2,a5              ;save the row end for later
Normal       move.l    -(a2),d0         ;get the next element
            jsr      _LVOSPDiv(a6)      ;divide by the diag. (in d1)
            move.l   d0,(a2)            ;store it back in the array
            dbra     d4,Normal          ;cont until row i normalized
                                        ;Note: a2 ptring to END of the
                                        ;row i-1
            move.l   -(a1),d0           ;get solution vector element
                                        ;corresponding to row i
            jsr      _LVOSPDiv(a6)      ;div by same diag element
            move.l   d0,(a1)            ;save it back in the vector

            move.l   SolTop,a0          ;the ops below will leave
                                        ;the prob sol in sol vec
            move.l   MatTop,a4          ;zero elements in ea. row
            move.l   Rank_1,d4          ;above/below diag. of row i
ElimCol      cmp.l     d4,d3            ;skip this if j-i
            beq      NextRow
            move.l   a5,a3              ;recover ptr to the end of
                                        ;row i
            move.l   0(a4,d2),d6        ;get diag. ele. of row j
            move.l   Rank_1,d5          ;do for ea.element of row j
FixRow       move.l    -(a3),d0         ;get element of row i to be
            move.l   d6,d1              ;Mult by diag. ele. of j
            jsr      _LVOSPMul(a6)      ;multiply them
            move.l   d0,d1              ;switch around the result
            move.l   -(a4),d0           ;get affected element row j
            jsr      _LVOSPSub(a6)      ;subtract our product
            move.l   d0,(a4)            ;save result in the matrix
            dbra     d5,FixRow          ;cont for elements of row j
            move.l   d6,d1              ;do the same mult. and sub.
            move.l   (a1),d0            ;on the solution vector ele.
            jsr      _LVOSPMul(a6)
            move.l   d0,d1
            move.l   -(a0),d0
            jsr      _LVOSPSub(a6)
            move.l   d0,(a0)
            bra      ReLoop             ;skip the rest
NextRow      suba.l    RowWidth,a4      ;if j-i then advance row j
            subq.l   #4,a0              ;ptr to the next row, j-1
ReLoop       dbra      d4,ElimCol
            dbra     d3,BigLoop         ;That's all for now
*
*       All done so print the answers
*
            move.l   _DOSBase,a6        ;back to writing
            WriteCon OutHand,#Msg5,#Msg5_Len
                                        ;print a line to this effect
            move.l   #Solution,a2       ;get start of the sol. vec.
            move.l   Rank_1,d5          ;print correct # of entries
SolLp2       move.l    #PBuf,-(SP)      ;push the buffer address
            move.l   d0,-(SP)           ;This does nothing, but is
                                        ;essential for op of _fpa
            move.l   (a2)+,-(SP)        ;push FFP -convert to string
            jsr      _fpa               ;call routine to convert it
            lea      12(SP),SP          ;clean up the stack
            move.w   lf,PBuf+14         ;add a lf to the buffer
            WriteCon OutHand,#PBuf,#16  ;print it out
            dbra     d5,SolLp2          ;continue until all printed
            bra      CloseShop          ;go to the end

DivBy0       move.l    _DOSBase,a6      ;Uh Oh! Gotta stop
            WriteCon OutHand,#Msg4,#Msg4_Len ;print the bad news
```



```
CloseShop    move.l    _AbsExecBase,a6  ;CloseLib in exec library
             move.l   _MathBase,a1      ;close the Math library
              jsr       _LVOCloseLibrary(a6) ;close it

NoMath        move.l    _AbsExecBase,a6
             move.l   _DOSBase,a1       ;close the dos library
              jsr       _LVOCloseLibrary(a6)

NoDOS          clr.l     d0             ;things went well, bye.
           rts

           SECTION  data,DATA
DOS_Name     dc.b       'dos.library',0
Mathffp_Name:
           dc.b      'mathffp.library',0
Msg1         dc.b  'Enter rank of the coefficient matrix',13,10
Msg1_Len     EQU         *-Msg1
Msg2         dc.b  'Enter the coefficients (row,col)',13,10
Msg2_Len     EQU         *-Msg2
Msg3          dc.b        'Enter the solution vector (row)',13,10
Msg3_Len     EQU         *-Msg3
Msg4         dc.b        'Oops.   Division by zero!',13,10
Msg4_Len     EQU         *-Msg4
Msg5         dc.b        'The unknowns are:',13,10
Msg5_Len     EQU         *-Msg5
lf             dc.w        $0D0A

           SECTION  mem,BSS
* variables
_DOSBase             ds.l        1
_MathBase            ds.l        1
InHand               ds.l        1
OutHand              ds.l        1
Rank                 ds.l        1
Rank_1               ds.l        1
RowWidth             ds.l        1
MatTop               ds.l        1
SolTop               ds.l        1
* buffers
RBuf                 ds.b        80
RBuf_Len  EQU          *-RBuf
PBuf                 ds.b        80
PBuf_Len      EQU          *-PBuf
Matrix               ds.l        81
Solution             ds.l        9
          END
```

•AC•

Do-It-yourself Improvements to the Amiga Genlock

by John Foust

If your Genlock has trouble synching with an external video signal, or if the video appears to tear away at the top edges, there is a simple fix is available.

By removing a single resistor from the circuit board, you can increase the quality of the genlock output. This change does violate the warranty of the genlock, so you must perform this fix at your own risk.

To make the alteration, you will need a small pair of pliers, a fine-nosed wire clipper and a screwdriver to remove the case of the genlock. First, unscrew the plastic case and remove the lid. Inside the genlock is a metal shield that protects the circuitry from interference. The metal shield is affixed by metal tabs. Grabbing these tabs with the pliers and twisting them back to a straightened position allows the shield to be removed from the case.

This handiwork exposes the circuit board. Each component on the circuit board is labelled on the surface of the board. An integrated circuit chip looks like a grey box about the width of a pencil. Find an integrated circuit labelled "U11"; it should have sixteen pins.

Now, you must locate a particular resistor. A resistor looks like a thin rod with colored stripes and wires jutting out of both ends. Our resistor is labelled "R55," but this note may be covered by the resistor itself. There is another way to find it. Near one end of integrated circuit U11 is a resistor with three colored bands yellow — violet and red. Search no more.

This resistor should be removed from the circuit board. Remove it – very carefully – using the fine-nosed wire clipper to clip one of the two wires at the end of the resistor. With the pliers, bend the resistor away from the circuit board and expose the wire at the other end. Clip this wire and the resistor should come free of the circuit board.

Be sure that no scraps of wire are left on the board. Replace the metal shield, twist the tabs back into position and screw the case back together.

• Thanks to Compuserve's Margaret Morrell, Roy Laufer, Louis Markoya and Ralph Landry for their comments.

•AC•

As I See It

WordPerfect, Gizmoz V2.0, and Zing!Keys

by Eddie Churchill

This article is not a review. It is also not the price index for the New York Stock Exchange. The opinions expressed within it are not those of PiM Publications, my parents (or anyone they know), or anyone I know for that matter. They are solely and completely my own. If they are your opinions as well, I'm very sorry but I had them first!

There are three new products that I've acquired this month that I want to talk about especially, because even if a review is done, it probably won't do them justice. And hey, a LITTLE repetition is good for the soul. But if I don't get organized, I'll lose track of where I am and not cover one, so excuse the outline mode at times.

WordPerfect

It came in this week; the industry standard word processor of the IBM world, now available for the Amiga. No longer will people look down their noses at I'm and say, "Yeah, its got swell graphics. When is there going to be some serious BUSINESS SOFTWARE for it, though?" (You've heard this tone of voice before. Its the one used by people who are politely looking at your favorite machine, but they're only doing it as a favor to you, because their minds are already made up.)

Of course, we've known all along that the software we had was quite adequate for business, thank you. But you can't tell anyone else that. So, we've waited. And look what shows up, and just in time for the 2000 too. (What a coincidence, huh?) WordPerfect for the Amiga. Four disks crammed full of everything that has made it one of the best selling word processors in a market that is FULL of competition. (Take a look at how many text editors/word processors there are for the IBM sometime. It will astound and amaze you. Why so many? How are they all different? Which one's best? How can the market support so many?)

And make no mistake about it, WordPerfect is a heavy contender in the IBM market. And there's good reason, too. Its loaded with features that are the type that you say "Wow! How did I live without this?" once you've tried it. (Those of you who have moved up from the IBM world know this feeling. Its the one you got after trying SideKick(tm) for the first time. We have a similar program. Its called Gizmoz(tm), and if you don't have it, you're crippling your productivity. It's THAT good.)

So, needless to say, WordPerfect was up and running before youcould say "WorkBench". Or, at least, it started to be. Perhaps I should say we started trying to get it up and running. The primary machine around here is a first run 1000 that's never given me a lick of trouble. Ken took it submarining with him one patrol about a year ago! Anyway, it only has 512K of memory. Only! Incredible, isn't it? A few years ago, 512K would have seemed like a MAINFRAME! Why, it took weeks just to fill a 170K floppy. How things change in the computer industry. Guess that's why I like it so much. Its so dynamic! Anywho, 512K is the BARE MINIMUM for WordPerfect. And you'd better have a T: assigned to something other than ram:, because there's not enough ram: there to help WP out at all. Better to have a T: directory on your second drive, where you'll be doing all your writing. That way there's enough room for all of WP's scratch files. You might think this was strange, since a cursory examination of the disk shows the program wp to be a measly 99,660 bytes long. That's not all that big. But, look in the Libs: directory! There's 106,584 bytes of overlay modules hiding in there, waiting to be called in to gobble up your 512K. So, the bottom line is, if you're using a 512K machine, reassign the T: directory to a disk with some room on it, so WordPerfect has some place to put its temporary files.

And you want those temporary files, believe me. They're your Undo buffer (replaces that last accidental deletion), your printer utility's spooler, and all your wonderful macros that are saving you SO much time. For those of you who haven't experienced the joys of macros, imagine being able to define the Amiga-A key to carry out about a dozen keystrokes that you repeat often. But what really makes macros great is redefining a three key sequence into one. THAT'S where you save the most time, I think. After all, those big complicated procedures are only done a couple of times a session. But the common, everyday things (like saving a document, cutting a section of text out, and so forth) usually require two or three keystrokes or mouse-events. There's the initial selection, then the requester for the name (same as what you loaded, so just hit return) then the requester asking if you want to replace the file that already exists with that name. Drag click, click, click. Just to save the file you're working on.

continued...

And you're trying to be conscientious and save your work often so that power glitch won't wipe out half an hour's work. It can get pretty irritating. That's when you are really glad for macros!

So, although you can run WordPerfect in a 512K machine with 2 drives, a Megabyte of memory or a hard drive is ALMOST a necessity. It screams right along on the 1 Meg A500 that came home with me a few days ago. (I wonder how long its going to be before a 1 Meg machine is considered the minimum system? The A500 certainly makes it feasible. With the 512K memory expansion for under $200, you just can't go wrong.) Still, I like my separate keyboard. I think there's plenty of room in the world for both machines (all three counting the A2000 too!), and the Amiga market can't help but improve with their becoming more and more popular.

But back to WordPerfect. What's so great about it?, you ask. Well, how about automatic outlining within a document? Yes, a normal text file, but right in the middle have an outline that is numbered (and renumbered when you add or delete an entry) automatically as you type. Looks professional, and sure makes those outlines more legible. How about a 110,000 word spelling checker, with a Thesaurus (I could never have spelled that without the checker!) available at a keystroke? User defined supplemental dictionary, of course. (although 110,000 is a lot of words? Who typed in all them suckers? I hope they got a nice vacation after all that. Can you imagine doing that? Paving, pavior, pavis, Pavlov... And don't you dare make a typo! Arghh!)

Another great thing about WordPerfect is that there is AT LEAST two ways to do EVERYTHING! There is a keystroke (sometimes several to get to the exact thing you want) for every feature and aspect. This is a real godsend for those of you who aren't all that fond of mice while doing word processing. I can see their point. If you're a fast typist, it probably would be a nuisance to take your hands off the keyboard to do your editing. With WP, that's not a problem. Not only do the mouse gadgets work on the requesters, but the first letter in whatever they say works, as well as return (for the default, which is always rendered so that you can tell which choice it is. Usually the gadget is thicker.) So there's usually three ways to tell a requester what you want to do. Pretty slick. We probably have the IBM world to thank for this, since the program was developed and went through 4 revisions there before it came to us. And the mouse is still not a very common creature attached to a PC. So, all those Alts and Ctrls have to be defined. And some people like that. Me? I'm just glad for the macros. And the spelling checker. And the repeat key. That's what the Esc key does for you. Hitting it brings up a requester for how many times you want the next command done. Then hit a command key or a macro! Now that's what I call productivity.

Hey! Here's good news for all of you who have used WordPerfect on an IBM or clone. The format is the same in both the IBM and Amiga versions. So, using DOS-2-DOS, you can bring any text files you did on a PC or clone right on over into WordPerfect on your Amiga, and all the formatting codes will still be right there, making your text look pretty. I know of a couple of folks who are glad for this, as it will make their transition from one machine to another LOADS easier. Especially people who travel, and need a computer with them. Until Commodore makes a Laptop, MS-DOS is the only game in town. And guess what runs on all those nifty little flat-screened wonders? Word Perfect Executive, that's what. So type away on that airplane, you'll be able to print it out on your Amiga when you get home.

Another nice feature is that there are a lot of defaults that the program assumes that you can change. All you have to do is boot the program from CLI typing "WP -s". This brings you into a set-up menu otherwise unavailable when the program is running. Here you can set how long between auto saves, (didn't I mention this? Oh, its great! Every 30 minutes the program saves my work todisk, whether I remember to or not.) how

your lines and pages are formatted and numbered (if at all), where the program is to look for the spelling checker (hard drive owners rejoice over this one, as well as people with more than 2 drives) and several other things I just can't think of right now. Even the default value in the repeat requester. Just great.

Gizmoz V2.0

Gizmoz was one of the first programs in the Desktop Accessories category to come out for the Amiga. It was great the second it hit the streets. It made up for some of the shortcomings of Workbench, filling in the holes that the gang at Los Gatos didn't have time to. The price was right and it wasn't copy protected. (I'm not sure, but it may very well have been the first piece of Amiga software not to have been.) Everything was available via icons for those who hadn't gotten the hang of CLI yet. In short, it was a super set of utilities that found its was into our hearts (and our disks) in a hurry.

Well, the best just got better! Gizmoz version 2.0 is out, and I love it more than I did the old one. Its not just that all the programs are smaller. Its not just that they've added several new utilities that are really great (I'll talk more about them in a little while). I think the thing that sold me on 2.0 was how POLISHED everything looked. This is obviously the work of people who are proud of their work, and who love the Amiga. It shows in every single program.

Take one of the calculators (there are three), it doesn't matter which one. They all worked fine in the old version. But now, not only are they smaller (code size wise), but when you press a number key on the keypad of your keyboard, the corresponding key on the screen is depressed. It looks so slick! Sort of like a player calculator. Its little touches like that I'm really impressed with. Just about anyone who calls himself a programmer (and even some who don't) can create a software calculator on the Amiga. It takes talent and taste to make one as neat as Digital Creations has.

And the calculators aren't the only nifty improvement in Gizmoz 2.0. Blackbook, the printer utility (still the only file printer that both listens to preferences for size of pitch and such AND correctly handles page breaks at that style) is even better than before. Now you don't even have to type in the name of the text file you want to print. Just click on the name field of the window, and a nice requestor with all the devices, directories, and files you have are only a couple of clicks away. They even made it so that it automatically open the file for printing if you pass it the filename from CLI. Really professional, guys.

I guess what I'm trying to get across is that Gizmoz is the SideKick(tm) of the Amiga. The programs are small, easily moved (thanks to the icons, which are always nicely done), and useful. Once you have this package, you never want to be without it again. There may be a Workbench disk around here that doesn't have at least one of the OVER 20 utilities from Gizmoz on it, but I can't find one right now. Heartily recommended.

Zing!Keys

Just when you thought that the macros in WordPerfect had improved your life to the utmost, along comes Zing!Keys and improves it again. This is another of those "can't live without once you get it" type packages. It is to Gizmoz what SuperKey(tm) was to SideKick(tm). Zing!Keys even makes working in CLI easier. If you can't (or just don't want to) cough up the big bucks for WordPerfect, Zing!Keys can breath new life into whatever word processor you are using.

And these are not your ordinary macros either. Not only can you record keystrokes, but mouse events (eventhe position of your mouse at any given time!) and Intuition events. Meridian Software has included a means to access certain Intuition functions (like the system time and date)

And macros are only a part of what makes Zing!Keys so exciting. They have included a screen saver and printer, both just a key press away. You can define Hot Keys (keys that

continued...

when pressed, start up other programs) to your heart's content. PopDPaint is my favorite. With just the press of Ctrl Alt D, Deluxe Paint][comes up and is running, no matter where I was or what I was doing before. (Needless to say, this is only happening on the A500. The 1000 with its measly 512K can barely keep Scribble! up and running with Zing!Keys loaded.)

And here we come to Zing!Key's only serious failing. Its the same complaint I had/have against Zing! itself (which is a real nice Alternate User Interface, by the way. More on that some other time.), and that is they take up too much room. Zing! takes up over 230K of space on your Workbench disk. I don't know about you, but I have a hard time getting 230K of free space on a Workbench disk now adays. Maybe someday when I get a hard-drive. Zing!Keys is, alas, no better.

Loading Zing!Keys will eat up 74,000 bytes in a flash. Not all that much, you say. But by the time you load workbench, a text editor (certainly not Wordperfect!) and a textfile, most 512K users will find that they are starting to scrape the bottom of the ole RAM barrel.

Oh, yes, one funny thing I noticed while installing Zing!Keys on a work disk for its trial run. Don't Runback it. (For those of you who don't know what it is; Runback is a Public Domain program that runs a program from CLI, but returns that CLI to you instead of just spawning a newcli to run the program in. Its a great little utility if you have lots of stuff in your startup-sequence, and hate seeing your CLI numbered 7> when you finally finish booting.) For some unknown reason, if you have "runback Zing!Keys" in your startup-sequence, when your CLI window disappears, you getto see the disk icons for about 2 seconds before they just blink out of existence. It looks funny, until you realize that you don't have a user interface anymore. Ack! Help! So be forewarned and don't do that. ("Doc, it hurts when I do this." "Then don't do that!")

With those slight caveats aside, however, Zing!Keys really is a great program. I'm hopelessly addicted to it. Its really neat to watch a macro you set up that actually moves the pointer. I have one that automatically clicks in the continue gadget in, say, the requester that comes up at the start of a diskcopy. Automation at its best.

Guess that about wraps it up for this month. Until new month, keep your keyboard clean, and remember "Piracy – Just say *NO!*"

•AC•

An AmigaBASIC™ File Browser

A full-featured browsing program offering vertical and horizontal scrolling via scroll bars and line or character-at-a-time gadgets, and MUCH MORE!

by Bryan Catley

Actually, not so basic at all, just written in BASIC! "The File Browsing Program" is a full-featured browsing program offering vertical and horizontal scrolling via scroll bars and line or character-at-a-time gadgets. A horizontal scale, general information about the file and three character string search modes: Top to bottom, current location to bottom, and wrap around (current location to current location) are also included.

The program is entirely menu driven, while also offering the option of using single key strokes in place of menu item selection. Since the file to be browsed is loaded into memory, the amount of free memory controls the maximum size file that may be browsed.

On a 512K Amiga with no other tasks running, and no additional routines loaded in memory (such as speech, cut and paste, etc), the maximum file that can be handled exceeds 150,000 bytes in length (although you will have to modify the CLEAR statement first); more than adequate for most files you will want to browse. Without modifying the CLEAR statement, you should be able to browse files up to about 50,000 bytes in length (Although, depending on the number of records in the file, you may have to use the File Size menu item to adjust the maximum number of records that can be handled).

Entering the Program

Entering the program is pretty straightforward. Just type it in, remembering to save a copy occasionally. Saving is especially important before you "Run" it for the first time. You never know what those typos may end up causing!

As shown, the program contains a number of comments which briefly describe the function of the piece of code that follows. Since these comments use memory, and since saving memory is important (especially with this program), you may choose to remove them once the program is working correctly. Be careful, though, because they do help you find your way around the program when you need to check something out.

Using the Program

When you "Run" the program, you will be presented with a title screen and instruction to use the menus to select the desired program function. There are three distinct menus, but only the first two may be selected at this time. The third menu is used exclusively when actually browsing a file. You will also notice that most menu items have a single letter (in parentheses) to their right. You may also select this menu item by pressing the indicated key. For example, you may "Quit" the program by selecting the "Quit" menu item or by pressing the "Q" key.

Since the program is menu driven, let's look at the three menus, and what the selection of each each item within them will result in.

First the Project menu:

Quit (Q) – Selection of Quit causes the program to terminate with control being passed back to Basic. This item may only be selected while the title screen is displayed.

Second, the Initialize menu:

File Size (S) – This menu item allows you to modify the size of the array which will contain the file to be browsed. When selected, you will be presented with a Requester which shows the existing value. *Note*: the value shown represents the maximum number of records that may presently be stored, NOT the maximum number of bytes. This occurs because the array must be DIMensioned to a number of records, rather than a number of bytes. Remember that the memory required by the array comes from the same area of memory which holds the program. Therefore, if you increase this value too much, you will also need to modify the CLEAR statement at the beginning of the program to provide additional memory. See the discussion on error conditions a little later for more information. This item may only be selected while the title screen is being displayed.

continued...

File Name (N) -- Select this item to specify the name of the file you wish to browse. You will be presented with a requester which requires up to three pieces of information: The disk drive (click in the appropriate box) – if omitted the current disk drive will be used. The path name (click in the box) – if omitted the current path is used. The file name (click in the box) – this piece of information is required.

At any time while entering information in this requester, you may click in any of the three gadgets across the bottom of the requester: Browse terminates the requester and starts the browse operation; Clear clears the requester, allowing you to start over; Quit returns you to the title screen. Also note that to clear the text box presently being used, just press the ESC key. This item may only be selected while the title screen is being displayed.

Third, and lastly, the Browse menu:

This menu only becomes available when the file to be browsed has been loaded into memory and is being displayed in the browse window.(See the next section for a discussion about the contents of the browse window).

Top (T)
The file is positioned so the first record is at the top of the display (Horizontal positioning is not affected).

Bottom (B)
The file is positioned so the last 18 records in the file are displayed (Horizontal positioning is not affected).

Look For (L)
Use this selection when it is necessary to initiate a search of the file for a specified character string. Once this option is selected, you will be presented with a requester, asking for the character string to be searched for. The string may consist of up to any 30 characters, including blanks. There are also two gadgets which you may click in at any time: OK starts the search using the selected search mode, while Cancel cancels the requester and returns you to the browse window.

When the string is found (using the current search mode), the requester is removed from the screen and the display changes so that the found string is placed in the upper left hand corner of the display. Should the string not be found, a message is presented in the requester, giving you a chance to either modify the string or cancel the request.

Repeat (R)
The previously specified character string is searched for again using the currently selected search mode (selected from the next three menu items).

Top to Bottom
When selected (indicated by a check mark) all character string searching will take place from the first record in the file to the last one, regardless of the current position within the file.

Current to End
When this item is selected (indicated by a check mark), all character string searching will take place from the current position within the file to the end of the file.

Wrap Around
When this item is selected (indicated by a check mark), all character string searching will take place from the current position within the file to its end. The search will then go back up to the start and back down to the current position.

Finished (F)
Browsing of the current file is terminated and you are returned to the title screen.

The Browse Window

Once the file to be browsed has been loaded into memory, you are presented with a screen which contains some basic information about the file, along with a display of the first 18 records in the file. The basic information consists of the name of the file (as you entered it), the number of records in the file, the length of the longest record and the record

number at the top of the display. A horizontal scale (1 to the maximum record length) and horizontal and vertical scroll bars are also available.

These scroll bars represent the maximum width and length of the file, while the white bar within them represents the current position of the window you see on the screen within the file.

To better understand this idea, imagine the file spread out flat upon the ground, with the window you see on the screen covering the upper left-hand corner of the file. Now, further imagine you can move this window from its present position, in a vertical direction, to a position which will place the last record in the file at the top of the display window. Further still, imagine you can also move the window horizontally to a position which places the last character in the longest record at the left edge of the display window. You now understand the constraints of the scroll bars in the display window!

To move around in the file display, just click in either scroll bar, and the display will change to reflect this new location. If you wish to "fine tune" the position of the display window within the file, just click in the arrow heads at either end of either scroll bar. This adjustment will cause the display to change either by a line at a time (vertically), or a character at a time (horizontally). Note the horizontal scale also automatically changes to reflect the new horizontal position.

One further thing which should be noted is that when the file is read into memory, an "End of File" record is added. Besides providing an obvious indication of the end of file position within the display, it also serves to show how many blank records are at the end of the file!

Error Conditions

As with most programs, you will run into a few error conditions. There is nothing wrong with error conditions, they are a fact of life. It is just necessary to know how to handle them! In fact, programs should be designed to handle as many error conditions as the programmer can anticipate occurring!

In the case of "The File Browsing Program," there are one or two situations you should know about.

First and foremost, you will be informed if you do not provide a file name, or if the name you provide cannot be found. In either case, you must supply a correct name or "Quit." Once the file name has been accepted, there are a couple of other hurdles to overcome.

First, if the size of the file will not fit in the available Basic memory, you will be informed. The only solution to this is to increase the memory specified in the CLEAR statement at the beginning of the program (You could modify the program to do this automatically, but then you risk an "Out of Memory" GURU visit, unless you get pretty fancy about the whole thing!)

Second, the array which holds the file may be DIMensioned at too small of a value. This means the array is simply not big enough to hold every record in the file. You can correct this situation by selecting "File Size (S)" from the Initialize Menu, and specifying a larger value.

If you get beyond this point, you should be OK! Go ahead and browse!

Programming Notes

Most of the programming in "The File Browsing Program" is pretty straightforward. If you've done any AmigaBasic programming at all, you shouldn't have too much trouble understanding how this program works. Just follow the logic carefully! The following comments may be of interest, however.

continued...

Colors

A program of this type needs to keep as much memory free as possible. In this case, it boiled down to using the Workbench screen and restricting the program to four colors. This limitation irked me a little bit because I like programs to be colorful! However, by changing the colors used in each new display, we are able to stick with four colors, remain colorful, and give the impression of using more than four! (Don't get carried away though).

Memory Conservation

This point sort of goes hand in hand with the last point. As indicated, we stick with the Workbench screen (and four colors) and we use non-smart-refresh windows (i.e. we redraw them after drawing something else over them). We use the same window for multiple displays, (the title screen and the browse window are the same Intuition window), and we draw our own message windows, over any existing window, rather than opening another Intuition Window.

Menu Selection or Keyboard Selection

The ability to select the next action via a menu selection OR a keyboard stroke is probably the most interesting thing about "File Browser" from a programming point-of-view. Many non-Basic programs offer this option, but it is very rarely seen in Basic programs. Sure, there is no easy way to use (and show) the Amiga-key/other-key combination which is typically used, but that does not mean that Basic programs must give up on this capability. However, they may have to settle for a slightly more mundane type of keyboard selection. Keeping this in mind, there are many programs that show examples of menu polling (WHILE mID=0:mID=MENU(0):WEND) and other programs that show examples of keyboard polling (WHILE key$="": key$=INKEY$: WEND), so why not combine them? Consider the following fragment of code:

```
mID=0:key$=""
WHILE mID=0 AND key$="":mId=MENU(0):key$=INKEY$:WEND
```

The biggest problem here is determining which of the two conditions was satisfied in order to leave the WHILE/WEND loop. That's really no problem at all! Just check mID for 0 or key$ for "", and if the the result is true, the other item was chosen by the user. Simple!

The next problem involves converting the selection to something which is common to both. The easiest thing is probably to convert the keyboard selection (if that is what was made) to a menu selection, and then to handle things just as if a menu selection had been made originally. As we'll see, this is another easy task!

If a menu selection is made, we just return the menu-id and menu-item as normal. However, if a keyboard selection is made, we must first ensure that it is a valid selection, and then convert it to a menu-id and item. To accomplish this, we set up an array with as many entries as there are menus, plus one. Then, just before waiting for the user to select something, we complete the array with the appropriate keys for each menu. Thus, entry one contains the valid keys for menu one, entry two for menu two, etc. We also store the number of menus involved in entry zero.

Now, if we ensure that the key values are stored in the same sequence as the menu items appear in the menu, it becomes a simple matter to convert a key stroke to a menu-id and item! The entry in the array represents the menu-id, while the position of each key within the entry represents the menu item.

There are, however, two things to watch out for. First, every valid keystroke must be unique. Second, we must be able to represent menu items which do not have associated key strokes. The easiest way of handling this requirement is to represent these keys with some character which is not likely to be entered by mistake. Almost any special character will do, in this case we use a $.

The sub-program GetNext, at the very end of the program, performs all these chores for "The File Browsing Program." It also does one other thing. Besides waiting on the menu

and keyboard, it also waits for a mouse click, indicated by the shared variable MouseInd becoming non-zero. In this event, it just exits without doing anything. This is necessary because while the user is actually browsing a file, he has three options: select a menu item, press a key or click in one of the scroll bars. We must be able to handle all three! Note how the subprogram uses the number of menus in entry zero to control its number of search loops. This feature ensures that GetNext may be used anywhere in any program to provide the user with this option.

I haven't even mentioned the three Gadgets and the GetIp subprograms because they have been fully described in earlier issues of *Amazing Computing*. If you are interested in reading more about these subprograms, you should check out the previous issues.

Well, I hope you will find "The File Browsing Program" both useful and a source of ideas for your own programs.

```
The File Browsing Program, Version 1.0
' ------------------
'
' Bryan D. Catley
' April 1987
'
' Increase the CLEAR ,70000& as instructed by the program.
CLEAR ,25000:CLEAR ,70000&
numbx=17:DispLen=76:MaxDisp=18:FileSize=500:Srch=1
type=0:ErrSw=0:MouseInd=0:NumRecs=0:MaxLen=0:Xposn=0:Yposn=0
EndRec=0:TxtCol=0:NewCur=0:n=0:x=0:m=0:mID=0:mItem=0:LkSt=0:Last=0
A%=0:B%=0:HorP%=0:VrtP%=0:LenHBar%=0:LenVBar%=0:offset%=0
offsetp%=0:MouseX%=0:MouseY%=0:TextLen%=0
Path$="":File$="":FullName$="":Dev$="":Scale$="":ErrMsg$=""
Text$="":TexTyp$="":SrchString$="":Prompt$=""
DIM bx(numbx-1,6),bxtxt$(numbx-1),MKeys$(3)
BldGadgets numbx,bx(),bxtxt$()
' File Information Gadgets (0-8)
DATA  86, 70,332, 12,1,3,-1,""
DATA  86, 86,172, 12,1,3,-1,""
DATA  46, 22, 12, 12,3,1, 2,""
DATA 152, 22, 12, 12,3,1, 2,""
DATA  46, 38, 12, 12,3,1, 2,""
DATA 152, 38, 12, 12,3,1, 2,""
DATA 120,108, 66, 16,3,1, 0,"Browse"
DATA 188,108, 62, 16,3,1, 0," Clear"
DATA 252,108, 56, 16,3,1, 0," Quit"
FIA%=0:FIB%=1:FIC%=2:FID%=8
' File Display Gadgets (9-14)
DATA   4, 24, 20,  7,0,0,-2,""
DATA  24, 24,560,  7,1,2,-2,""
DATA 592, 24, 20,  7,0,0,-2,""
DATA 612, 40, 12,  8,0,0,-2,""
DATA 612, 48, 12,128,1,2,-2,""
DATA 612,176, 12,  8,0,0,-2,""
DIA%=9:DIB%=14
' Requester Gadgets (15-16)
DATA 228,124, 56, 16,3,0, 0,"  OK"
DATA 354,124, 56, 16,3,0, 0,"Cancel"
RQA%=15:RQB%=16
HorStX%=bx(10,0):HorStY%=bx(10,1):HorLen%=bx(10,2)
VrtStX%=bx(13,0):VrtStY%=bx(13,1):VrtLen%=bx(13,3)
ON MOUSE GOSUB GetMouse
MENU 1,0,1,"Project"
MENU 1,1,1," Quit (Q)"
MENU 2,0,1,"Initialize"
MENU 2,1,1," File Size (S)"
MENU 2,2,1," File Name (N)"
MENU 3,0,0,"Browse"
MENU 3,1,1," Top          (T)"
MENU 3,2,1," Bottom       (B)"
MENU 3,3,1," Look For     (L)"
MENU 3,4,0," Repeat       (R)"
MENU 3,5,2,"  Top to Bottom "
MENU 3,6,1,"  Current to End"
MENU 3,7,1,"  Wrap Around   "
MENU 3,8,1," Finished     (F)"
MENU 4,0,0,""
WINDOW 2,,,0,-1
Main:
DoTitle
MENU 1,0,1:MENU 2,0,1:MENU 3,0,0
```



```
COLOR Mag,Blk
LOCATE 10,38:PRINT"THE"
LOCATE 12,36:PRINT"F I L E"
LOCATE 14,32:PRINT"B R O W S I N G"
LOCATE 16,36:PRINT"PROGRAM"
COLOR Yel,Blk:LOCATE 21,23
PRINT"Use Menus to select program function"
MKeys$(0)="2":MKeys$(1)="Q":MKeys$(2)="SN"
GetNext MKeys$(),mID,mItem
ON mID GOTO ProjMenu,InitMenu

' Mouse Event Routine
' -------------
GetMouse:
GetGadget A%,B%,bx(),bxtxt$(),type:RETURN

' File Open Error Routine
' -------------
OpenError:
WINDOW 2:DoTitle
WINDOW 3:GOSUB SetWind3
DrawGadgets FIA%,FIB%,bx(),bxtxt$()
COLOR Blk,Cyn:LOCATE 8,18:PRINT LEFT$(Dev$,3):COLOR Blk,Blu
LOCATE 10,12:PRINT Path$
LOCATE 12,12:PRINT File$
IF ERR=53 THEN
  ErrMsg$="File Not Found"
ELSE
  ErrMsg$="Open Error:"+STR$(ERR)
END IF
ErrSw=1:RESUME AfterOpen

' Title Screen Active Menus
' -------------
ProjMenu:ON mItem GOTO Quit
InitMenu:ON mItem GOTO SetSize,FileInfo

' Change Maximum Size of File
' -------------
SetSize:
Prompt$="Enter New File Size:"
Text$=STR$(FileSize):Text$=RIGHT$(Text$,LEN(Text$)-1)
TexTyp$="INT":TextLen%=4:GOSUB Ask
ON type GOTO NewSize,Main
NewSize:
```

continued...

```
FileSize=VAL(Text$):GOTO Main

' Get Information About File to Browse
' ------------------
FileInfo:
WINDOW 3,"Browse Information:",(100,20)-(540,156),0,-1
MENU 1,0,0:MENU 2,0,0:MENU 3,0,0:GOSUB SetWind3
FItext:
DrawGadgets FIA%,FIB%,bx(),bxtxt$():COLOR Blk,Blu
LOCATE 10,12:PRINT Path$
LOCATE 12,12:PRINT File$
LINE(216,2)-STEP(176,36),Cyn,bf
TxtCol=Blk:A%=FIA%:B%=FID%:MOUSE ON
FIwait:
type=0:WHILE type=0:SLEEP:WEND
FIgo:
ON type GOTO FIpath,FIname,df0,df2,df1,dh1,FIstart,FIclear,FIquit
FIpath:
COLOR Blk,Wht:LOCATE 10,12::GetIp Path$,"CHAR",40
IF Path$<>"" AND RIGHT$(Path$,1)<>"/" THEN Path$=Path$+"/"
IF MouseInd=0 THEN FIwait ELSE FIgo
FIname:
COLOR Blk,Wht:LOCATE 12,12:GetIp File$,"CHAR",20
IF MouseInd=0 THEN FIwait ELSE FIgo
df0:Dev$="DF0:":GOTO PrtDev
df1:Dev$="DF1:":GOTO PrtDev
dh1:Dev$="DH1:":GOTO PrtDev
df2:Dev$="DF2:":GOTO PrtDev
PrtDev:
COLOR Blk,Cyn:LOCATE 8,18:PRINT LEFT$(Dev$,3):COLOR Blk,Blu
GOTO FIwait
FIclear:
Path$="":File$="":GOTO FItext
FIquit:
MOUSE OFF:WINDOW CLOSE 3:GOTO Main

' Check Out the Specified File
' ------------------
FIstart:
MOUSE OFF
IF File$="" THEN
  BEEP
  LINE(222,6)-STEP(164,28),Wht,bf
  LINE(222,6)-STEP(164,28),Blu,B
  LINE(224,8)-STEP(160,24),Blu,B
  COLOR Blu,Wht:LOCATE 3,33:PRINT"No File Name"
  MOUSE ON:GOTO FIwait
END IF
FullName$=Dev$+Path$+File$
ON ERROR GOTO OpenError
OPEN FullName$ FOR INPUT AS #1
AfterOpen:
ON ERROR GOTO 0
IF ErrSw=1 THEN
  ErrSw=0:BEEP
  LINE(222,6)-STEP(164,28),Wht,bf
  LINE(222,6)-STEP(164,28),Blu,B
  LINE(224,8)-STEP(160,24),Blu,B
  COLOR Blu,Wht:LOCATE 3,32:PRINT ErrMsg$
  MOUSE ON:GOTO FIwait
END IF
WINDOW CLOSE 3:WINDOW 2

' Load File into Memory & Calculate Various Values
' ------------------
PALETTE 0,0,0,0:Blk=0
PALETTE 1,1,0,1:Mag=1
PALETTE 2,.8,.8,.8:Gra=2
PALETTE 3,1,.8,0:Yel=3
COLOR Mag,Blk:CLS
LOCATE 2,1:PRINT"File: ";
COLOR Yel,Blk:PRINT FullName$
LOCATE 24,23:PRINT"==> The File Browsing Program <==";
DrawGadgets DIA%,DIB%,bx(),bxtxt$():COLOR Gra
AREA(4,28):AREA STEP(22,-4):AREA STEP(0,7):AREAFILL
AREA(584,24):AREA STEP(22,4):AREA STEP(-22,3):AREAFILL
AREA(612,48):AREA STEP(6,-8):AREA STEP(6,8):AREAFILL
AREA(612,176):AREA STEP(12,0):AREA STEP(-6,8):AREAFILL
LINE(0,40)-STEP(608,143),Gra,bf
IF (LOF(1)+1000)>FRE(0) THEN NoMem
COLOR Mag,Gra:LOCATE 8,33:PRINT"Loading File"
DIM Records$(FileSize)
NumRecs=0:MaxLen=0
WHILE NOT EOF(1) AND NumRecs<FileSize
  LINE INPUT #1,Records$(NumRecs)
  IF LEN(Records$(NumRecs))>MaxLen THEN MaxLen=LEN(Records$(NumRecs))
  NumRecs=NumRecs+1
WEND
CLOSE #1
IF NumRecs=FileSize THEN TooBig
Records$(NumRecs)="* * *  End of File  * * *":NumRecs=NumRecs+1
FOR n=1 TO (NumRecs/10)+1
  IF n<10 THEN x=8 ELSE x=7
  Scale$=Scale$+LEFT$("----+----",x)
  Scale$=Scale$+RIGHT$(STR$(n*10),LEN(STR$(n*10))-1)
NEXT
IF MaxLen<LEN(Scale$) THEN Scale$=MID$(Scale$,1,MaxLen)
HorP%=CINT((DispLen/(MaxLen+DispLen))*100)
LenHBar%=(HorLen%/100)*HorP%
VrtP%=CINT((MaxDisp/(NumRecs+MaxDisp))*100)
LenVBar%=(VrtLen%/100)*VrtP%
LOCATE 8,33:PRINT SPACE$(12)
MENU 1,0,0:MENU 2,0,0:MENU 3,0,1

' Display File
' ------------
COLOR Mag,Blk
LOCATE 3,1:PRINT USING"Number of Records in File: ###";NumRecs-1;
PRINT USING"; Longest is: ###";MaxLen;:PRINT"; Record at Top is: "
Xposn=1:Yposn=1
GOSUB DoHorInfo:GOSUB DoVrtInfo
DispRecs:
LINE(0,40)-STEP(608,143),Gra,bf
COLOR Blk,Gra:LOCATE 6,1
FOR n=Yposn-1 TO EndRec-1
  PRINT MID$(Records$(n),Xposn,DispLen)
NEXT
DispWait:
MKeys$(0)="2":MKeys$(1)="Q":MKeys$(2)="TBLR$$$F"
A%=DIA%:B%=DIB%:type=0
MOUSE ON:GetNext MKeys$(),mID,mItem:MOUSE OFF
IF MouseInd<>0 THEN
  ON type GOTO Left1,HMove,Right1,Up1,VMove,Down1
ELSE
  ON mID GOTO ProjMenu,BrowseMenu,BrowseMenu
END IF
Left1:
IF Xposn=1 THEN DispWait
Xposn=Xposn-1:GOTO HDispRecs
HMove:
offset%=MouseX%-(HorStX%-1):offsetp%=CINT((offset%/HorLen%)*100)
Xposn=INT((MaxLen/100)*offsetp%)
IF Xposn<1 THEN Xposn=1
GOTO HDispRecs
Right1:
IF Xposn=MaxLen THEN DispWait
Xposn=Xposn+1:GOTO HDispRecs
Up1:
IF Yposn=1 THEN DispWait
Yposn=Yposn-1:GOTO VDispRecs
VMove:
```

```
offset%=MouseY%-(VrtStY%-1):offsetp%=CINT((offset%/VrtLen%)*100)
Yposn=INT((NumRecs/100)*offsetp%)
IF Yposn<1 THEN Yposn=1
GOTO VDispRecs
Down1:
IF Yposn=NumRecs THEN DispWait
Yposn=Yposn+1:GOTO VDispRecs
HDispRecs:
GOSUB DoHorInfo:GOTO DispRecs
VDispRecs:
GOSUB DoVrtInfo:GOTO DispRecs

' Display Horizontal Information
' ------------------
DoHorInfo:
COLOR Mag,Blk:LOCATE 5,1:PRINT SPACE$(DispLen)
LOCATE 5,1:PRINT MID$(Scale$,Xposn,DispLen);
offsetp%=CINT((Xposn/MaxLen)*100)
offset%=CINT(((HorLen%-LenHBar%)/100)*offsetp%)
LINE(HorStX%+1,HorStY%+1)-STEP(HorLen%-1,5),Mag,bf
LINE(HorStX%+offset%-2,HorStY%+2)-STEP(LenHBar%,3),Gra,bf
RETURN

' Display Vertical Information
' ----------------
DoVrtInfo:
COLOR Mag,Blk:LOCATE 3,68:PRINT USING "###";Yposn;
IF NumRecs<MaxDisp THEN EndRec=NumRecs ELSE EndRec=Yposn+MaxDisp-1
IF EndRec>NumRecs THEN EndRec=NumRecs
offsetp%=CINT((Yposn/NumRecs)*100)
offset%=CINT(((VrtLen%-LenVBar%)/100)*offsetp%)
LINE(VrtStX%+2,VrtStY%+1)-STEP(7,VrtLen%-2),Mag,bf
LINE(VrtStX%+3,VrtStY%+offset%)-STEP(6,LenVBar%),Gra,bf
RETURN

' Browse Menu Items
' ------------
BrowseMenu:
ErrMsg$=""
ON mItem GOTO GoToTop,GoToBot,LookFor,Repeat,Look1,Look2,Look3,Done

' Move to Top or Bottom of File
' ----------------
GoToTop:
IF Yposn=1 THEN DispWait
Yposn=1:GOTO VDispRecs
GoToBot:
IF NumRecs<=MaxDisp THEN DispWait
Yposn=NumRecs-MaxDisp+1:GOTO VDispRecs

' Determine what to Look For
' ----------------
LookFor:
Prompt$="Look For:":Text$=SrchString$:TexTyp$="CHAR":TextLen%=30
MENU 3,2,0:GOSUB Ask
ON type GOTO DoLook,DispRecs
DoLook:
SrchString$=Text$:COLOR Mag,Yel:LOCATE 15,32:PRINT SPACE$(16)
LOCATE 15,37:PRINT"Looking":MENU 3,2,1
Repeat:
IF Srch=1 THEN LkSt=0 ELSE LkSt=Yposn
Last=NumRecs-1:x=0:GOSUB Search
IF x=0 AND Srch=3 THEN LkSt=0:Last=Yposn-1:GOSUB Search
IF x<>0 THEN
  Yposn=n+1:Xposn=x
  GOSUB DoHorInfo:GOSUB DoVrtInfo
  GOTO DispRecs
ELSE
  ErrMsg$="String not found"
  GOTO LookFor
END IF
Search:
FOR m=LkSt TO Last
  x=INSTR(Records$(m),SrchString$):IF x<>0 THEN n=m:m=Last
NEXT
RETURN

' Set Up Type of Search
' ------------
Look1:MENU 3,3,2:MENU 3,4,1:MENU 3,5,1:Srch=1:GOTO DispWait
Look2:MENU 3,3,1:MENU 3,4,2:MENU 3,5,1:Srch=2:GOTO DispWait
Look3:MENU 3,3,1:MENU 3,4,1:MENU 3,5,2:Srch=3:GOTO DispWait

' File is too Big for Array as Dimensioned
' ----------------------
TooBig:
LINE(214,70)-STEP(180,68),Mag,B:LINE(216,72)-STEP(176,64),Mag,bf
COLOR Blk,Mag:LOCATE 11,32:PRINT"File too large."
LOCATE 13,31:PRINT"Adjust File Size"
LOCATE 14,33:PRINT"& try again."
LOCATE 16,30:PRINT"Click to continue"
WHILE MOUSE(0)=0:WEND:GOTO Done

' Not enough Memory
' ------------
NoMem:
```



```
LINE(214,70)-STEP(180,68),Mag,B:LINE(216,72)-STEP(176,64),Mag,bf
COLOR Blk,Mag:LOCATE 11,30:PRINT"Not enough memory."
n=LOF(1)-FRE(0)+1000
LOCATE 12,30:PRINT USING"Add #####";n;:PRINT" to CLEAR"
LOCATE 13,30:PRINT"at program start."
LOCATE 16,30:PRINT"Click to continue"
WHILE MOUSE(0)=0:WEND:GOTO Main

' Finished Browsing this File
' ------------
Done:
ERASE Records$:GOTO Main

' Time to Quit and Return to Basic
' ----------------
Quit:
WINDOW CLOSE 2:MENU RESET
PALETTE 0,0,.3,.6:PALETTE 1,1,1,1
PALETTE 2,0,0,.1:PALETTE 3,1,.5,0
CLEAR ,25000:END

' Draw Window 3
' ------
SetWind3:
PALETTE 0,0,0,0 :Blk=0
PALETTE 1,0,0,1 :Blu=1
PALETTE 2,0,1,1 :Cyn=2
PALETTE 3,1,1,1 :Wht=3
COLOR Blk,Cyn:CLS
LOCATE 2,2:PRINT"Select Disk Drive:"
LOCATE 4,2:PRINT"DF0:          DF2:"
LOCATE 6,2:PRINT"DF1:          DH1:"
LOCATE 8,2:PRINT"Drive Selected:"
LOCATE 10,2:PRINT"Pathname:"
LOCATE 12,2:PRINT"Filename:"
COLOR Blu,Cyn:LOCATE 6,29:PRINT"Click in Desired Box"
COLOR Blk,Blu:DrawGadgets FIC%,FID%,bx(),bxtxt$()
RETURN

' Set Up a Requester in the Current Window
' ----------------
Ask:
LINE(186,70)-STEP(268,76),Yel,B
LINE(188,72)-STEP(264,72),Yel,bf
LINE(196,94)-STEP(252,10),Blk,B
```

continued...

```
COLOR Blk,Yel:LOCATE 11,26:PRINT Prompt$
DrawGadgets RQA%,RQB%,bx(),bxtxt$()
COLOR Mag,Yel:LOCATE 15,32:PRINT ErrMsg$
MOUSE ON
GetIt:
A%=RQA%:B%=RQB%:type=0:TxtCol=Mag
LOCATE 13,26:GetIp Text$,TexTyp$,TextLen%
IF Text$="" THEN BEEP:GOTO GetIt
IF MouseInd=0 THEN type=1
MOUSE OFF:RETURN

' Various Subprograms
' ----------
SUB DoTitle STATIC
SHARED Blk,Blu,Mag,Yel
PALETTE 0,0,0,0 :Blk=0
PALETTE 1,1,0,0 :Red=1
PALETTE 2,1,0,1 :Mag=2
PALETTE 3,1,.8,0:Yel=3
COLOR ,Blk:CLS
AREA(376,8):AREA STEP(64,0):AREA STEP(-20,16)
AREA STEP(0,24):AREA STEP(-24,0):AREA STEP(0,-24)
COLOR Yel:AREAFILL
AREA(360,8):AREA STEP(32,0):AREA STEP(0,12)
AREA STEP(-16,0):AREA STEP(0,4):AREA STEP(8,0):AREA STEP(0,8)
AREA STEP(-8,0):AREA STEP(0,4):AREA STEP(24,0):AREA STEP(0,12)
AREA STEP(-40,0):COLOR Mag:AREAFILL
AREA(328,8):AREA STEP(24,0):AREA STEP(0,28)
AREA STEP(24,0):AREA STEP(0,12):AREA STEP(-48,0)
COLOR Red:AREAFILL
AREA(272,8):AREA STEP(64,0):AREA STEP(0,12)
AREA STEP(-20,0):AREA STEP(0,28):AREA STEP(-24,0):AREA STEP(0,-28)
AREA STEP(-20,0):COLOR Yel:AREAFILL
AREA(264,8):AREA STEP(16,0):AREA STEP(24,40)
AREA STEP(-16,0):AREA STEP(-8,-12):AREA STEP(-16,0):AREA STEP(-8,12)
AREA STEP(-16,0):COLOR Mag:AREAFILL
AREA(200,8):AREA STEP(56,0):AREA STEP(0,16)
AREA STEP(-24,0):AREA STEP(0,-4):AREA STEP(-8,0):AREA STEP(0,16)
AREA STEP(8,0):AREA STEP(0,-4):AREA STEP(24,0):AREA STEP(0,16)
AREA STEP(-56,0):COLOR Red:AREAFILL
COLOR Red,Blk:LOCATE 24,7
PRINT"Bryan D. Catley  2221 Glasgow Road  Alexandria  Virginia  22307-
1819";
END SUB

SUB BldGadgets (Num,t1(),t2$()) STATIC
FOR n=0 TO Num-1
  FOR m=0 TO 6
    READ t1(n,m)
  NEXT m
  READ t2$(n)
NEXT n
END SUB

SUB DrawGadgets (Ga%,Gb%,t1(),t2$()) STATIC
FOR n=Ga% TO Gb%
  x1=t1(n,0):y1=t1(n,1):x2=x1+t1(n,2):y2=y1+t1(n,3)
  bg=t1(n,4):fg=t1(n,5):bo=t1(n,6)
  LINE(x1,y1)-(x2,y2),bg,bf:LINE(x1,y1)-(x2,y2),fg,B
  IF bo>-1 THEN
    LINE(x1+2,y1+2)-(x2-2,y2-2),fg,B
    LINE(x2+1,y1+1)-(x2+1,y2+1),bo
    LINE(x2+1,y2+1)-(x1+1,y2+1),bo
    COLOR fg,bg:row%=INT(y1/8+2):col%=INT(x1/8+2)
    LOCATE row%,col%:PRINT t2$(n)
  END IF
NEXT n
END SUB

SUB GetGadget (Ga%,Gb%,t1(),t2$(),type) STATIC
SHARED MouseX%,MouseY%,MouseInd
WHILE MOUSE(0)=0:WEND
r%=CSRLIN:c%=POS(0)
mx=MOUSE(1):my=MOUSE(2)
MouseX%=mx:MouseY%=my:MouseInd=0
FOR n=Ga% TO Gb%
  IF mx>t1(n,0) AND mx<t1(n,0)+t1(n,2) THEN
    IF my>t1(n,1) AND my<t1(n,1)+t1(n,3) THEN
      bg=t1(n,4):fg=t1(n,5):bo=t1(n,6)
      IF bo>-1 THEN
        x1=t1(n,0)+2:y1=t1(n,1)+2
        x2=x1+t1(n,2)-4:y2=y1+t1(n,3)-4
        LINE(x1,y1)-(x2,y2),fg,bf
        COLOR bg,fg:row%=INT(y1/8+2):col%=INT(x1/8+2)
        LOCATE row%,col%:PRINT t2$(n)
      ELSE
        IF bo=-1 THEN
          x1=t1(n,0):y1=t1(n,1):x2=x1+t1(n,2):y2=y1+t1(n,3)
          LINE(x1,y1)-(x2,y2),fg,bf:LINE(x1,y1)-(x2,y2),bg,B
        END IF
      END IF
      type=n-Ga%+1:n=Gb%:MouseInd=1
      IF bo>-1 THEN n%=type+Ga%-1
    END IF
  END IF
NEXT n
WHILE MOUSE(0)<>0:WEND
IF type<>0 AND bo>-1 THEN DrawGadgets n%,n%,t1(),t2$()
LOCATE r%,c%
END SUB

SUB GetIp (Text$,DataType$,MaxLen%) STATIC
SHARED TxtCol,NewCur,MouseInd
start=POS(0):Cur=0:COLOR TxtCol
xpix=(start-1)*8:ypix=(CSRLIN-1)*8
IF FirstTime=0 THEN FirstTime=1:NewCur=1:DIM IPcursor%(46)
IF NewCur=1 THEN
  NewCur=0
  CurCol=TxtCol-1:IF CurCol<0 THEN CurCol=TxtCol+1
  LINE(xpix,ypix)-STEP(7,7),CurCol,bf
  GET(xpix,ypix)-STEP(7,7),IPcursor%
END IF
ShoText:
GOSUB DisplayText
NxtChar:
x$="":MouseInd=0:LeftPart$="":RightPart$=""
WHILE x$="" AND MouseInd=0:x$=INKEY$:WEND
IF MouseInd<>0  THEN GetDone  ' Mouse was clicked
IF x$=CHR$(30)  THEN CurRight ' Right-cursor
IF x$=CHR$(31)  THEN CurLeft  ' Left-cursor
IF x$=CHR$(8)   THEN DelLeft  ' Back-space key
IF x$=CHR$(127) THEN DelRight ' Delete key
IF x$=CHR$(27)  THEN ClrText  ' Escape key
IF x$=CHR$(13)  THEN GetDone  ' Return key
IF DataType$="CHAR" THEN
  IF x$<CHR$(32) OR x$>CHR$(127) THEN
    BEEP:GOTO NxtChar
  END IF
ELSEIF DataType$="REAL" THEN
  IF (x$<CHR$(48) OR x$>CHR$(57)) AND (x$<>".") THEN
    BEEP:GOTO NxtChar
  END IF
ELSEIF DataType$="INT" THEN
  IF x$<CHR$(48) OR x$>CHR$(57) THEN
    BEEP:GOTO NxtChar
  END IF
END IF
InsertChar:
IF LEN(Text$)=MaxLen% THEN BEEP:GOTO NxtChar
IF Cur>0 THEN LeftPart$=MID$(Text$,1,Cur)
IF LEN(Text$)>0 THEN RightPart$=MID$(Text$,Cur+1,LEN(Text$)-
LEN(LeftPart$))
Text$=LeftPart$+x$+RightPart$:Cur=Cur+1
GOTO ShoText
CurRight:
IF Cur=LEN(Text$) THEN NxtChar
Cur=Cur+1:GOTO ShoText
CurLeft:
IF Cur=0 THEN NxtChar
Cur=Cur-1:GOTO ShoText
DelLeft:
IF LEN(Text$)=0 OR Cur=0 THEN BEEP:GOTO NxtChar
IF Cur>1 THEN LeftPart$=MID$(Text$,1,Cur-1)
IF LEN(Text$)>Cur THEN RightPart$=MID$(Text$,Cur+1,LEN(Text$)-Cur)
Text$=LeftPart$+RightPart$
Cur=Cur-1:GOTO ShoText
DelRight:
IF LEN(Text$)=0 OR Cur=LEN(Text$) THEN BEEP:GOTO NxtChar
IF Cur>0 THEN LeftPart$=MID$(Text$,1,Cur)
IF Cur+1<LEN(Text$) THEN RightPart$=MID$(Text$,Cur+2,LEN(Text$)-Cur+1)
Text$=LeftPart$+RightPart$
GOTO ShoText
ClrText:
PRINT SPACE$(MaxLen%+1);:LOCATE ,start
Cur=0:Text$="":GOTO ShoText
DisplayText:
PRINT Text$+SPACE$(MaxLen%+1-LEN(Text$));:LOCATE ,start
xpix=(start+Cur-1)*8:PUT(xpix,ypix),IPcursor%
RETURN
GetDone:
PUT(xpix,ypix),IPcursor%
END SUB

SUB GetNext (MKeys$(),mID,mItem) STATIC
SHARED MouseInd
Key$="":mID=0:mItem=0:MouseInd=0:n=0:m=0
NumMenus=VAL(MKeys$(0))
WHILE Key$="" AND mID=0 AND MouseInd=0
  Key$=INKEY$:mID=MENU(0)
  IF Key$<>"" THEN
    FOR n=1 TO NumMenus
      m=INSTR(MKeys$(n),UCASE$(Key$))
      IF m<>0 THEN mID=n:mItem=m:n=NumMenus
    NEXT
    Key$=""
  ELSE
    IF mID<>0 THEN mItem=MENU(1)
  END IF
WEND
END SUB
```

•AC•

Roomers

Psssst! Did ya' hear about....

by the Bandito

The Amiga 2000 is now shipping to dealers, and users are exploring the new hardware and software. The Commodore hard disk controller performs very well in early benchmarks, surpassing even the PAL Jr. hard disk. Commodore is very excited about networking the Amiga 2000, and has bought dozens of network cards from Ameristar.

The new Amiga 500 and 2000 technical manuals drop a few hints about improved graphics chips, and warn developers about pitfalls to avoid when developing new hardware for these machines. The hints indicate more graphics CHIP memory, as well as more colors available on the screen at once.

One meg in beta

A late-breaking rumor claims the one-megabyte Fatter Agnes graphics chip has entered beta testing at select Amiga software developers. The chip is expected to be a direct replacement for the current Fat Agnes chip.

Rumors persist about high-end graphics cards in development in West Chester, including a monochrome high-resolution board aimed at CAD applications, and another board with more colors. Commodore is also reportedly working on a de-interlacing board for the 2000 video slot that works with NEC Multisync monitors.

Sources within Commodore say the New York Institute of Technology frame grabber and genlock are still months away from completion. Problems in hardware, software and production are the cause of the delay. This board is made for the Amiga 2000 only.

Amiga 2000 laser printer

Another great rumor is a low-cost laser printer for the Amiga 2000. It will have some memory and smarts on a expansion card, and uses a cheap laser engine as the printer. According to one source, it will be bundled with an Amiga 2000 in a special promotion this fall. Laser printers have appeared in the Commodore booths at shows for the last

Faster printing

Work continues on AmigaDOS 1.3. Work is being done to improve the printer driver system. The "printer.device" is being made faster, and more printer drivers are being created. Programmers are discussing improvements to the rendering of color images, too. No word yet on how these improvements will reach consumers.

It's obvious Commodore wants to sell a lot of machines before the end of the year – they even plan to advertise! They went as far as sending dealers a list of scheduled television, radio and magazine ads. Keep an eye peeled this fall. Commodore's ad agency was working on an animated commercial for the Amiga 500. It reportedly has Amiga 500s on pedestals breaking out of the ground, each running different software, such as the rotating head of Zeus, the Juggler, and animations from VideoScape and Animator: Apprentice.

Video news

The first run of Amiga Live! has come off the production line, for a total of 500 units. According to current gossip, they are still waiting for the plastic cases to come in from Japan.

The Bandito predicts that the absence of real-time video digitizers will end in the next six months, and that there will be not one but as many as eight contenders in the video digitizer and frame buffer market.

Here's your chance to play Bandito: There are at least three obvious contenders in the video hardware market, companies with existing or announced products. At least three other companies have leaked plans for video hardware in public places. And two remain hidden, so far. All hope to ship in the next few months. At least one will premiere at the AmiExpo show in New York in October.

The Amiga 500 software bundling plan has moved a lot of software for a few select companies. Developers not included in the plan aren't so happy. After all, if someone buys Word Perfect for next to nothing, will they buy another normally less-expensive word processor?

Developers who were included in the plan aren't so happy, either. Rumors say that Commodore only paid developers six percent over their cost of the software packaging. Both groups have referred to the plan as the "Screw the developers policy."

continued...

There has been hot traffic in the coupons for the software bundling, too. Commodore 64 owners were sent a coupon to get an Amiga 500 for a reduced price, bundled with about $1000 worth of software. Adertisements appeared in the newspapers of some cities. Dealers had placed the ads to buy coupons for a small amount from Commodore 64 owners, which they then gave to prospective buyers of the Amiga 500 who entered their store.

Grey market

Rumors continue to flow about the existence of a grey market for Amiga 1000 computers. Dealers want to buy the serial numbers of Amiga 1000 machines so they can sell Amiga 2000s at a reduced price by selling someone an Amiga 1000, never letting them take it from the store, then immediately letting the customer trade it in for an Amiga 2000. The customer gets a lower price, the dealer gets a merchandise credit from Commodore.

Los Gatos lives

Former Commodore-Amiga programmer Dale Luck was signed another 3-month contract to continue work on AmigaDOS in the tiny West Coast offices. Bob "Kodiak" Burns has signed a part-time contract, working with Luck on Amiga software.

New developer conference

The next Amiga developer conference may be held later this year in Philadelphia, close to the West Chester offices. Reportedly, Commodore managers want West Chester technicians to be able to drive home at night, to reduce travel and lodging expenses. Also, most Amiga events have been held on the West Coast, and East Coast developers want one closer to home.

Commodore technical support has cracked down on developers who haven't officially registered. The full developer registration cost is $450. If Commodore has 1000 developers, they made a half-million from fees alone. Some well-known Amiga companies haven't ever bothered to register. Some were told they could not appear in the Commodore booth at the next show unless they paid their $450.

Amiga in media

A spy within the CBS record offices in New York reports that the synthesizers on the upcoming "Tunnel of Love" album by Bruce Springsteen were controlled by an Amiga 500. Look for Amiga graphics on a new syndicated game show called Lingo. The host of the show is Michael Reagan, the President's son. Amiga graphics from VideoScape 3D will be used in a Campbell's soup commercial shown during the World Series.

•AC•

Modula-2 Programming on the Amiga™

Devices, I/O, and the Serial Port

by Steve Faiwiszewski

In this installment, I would like to discuss devices and I/O, and focus on the Serial Device. I would also present a module of general purpose serial port routines. But first, here is a crash course on strange and magical items known as Exec, messages and ports.

Exec

The heart (or rather the brain) of the Amiga's multitasking ability is something called the Exec (for multitasking executive). This part of the ROM Kernel routines is responsible for managing tasks, interrupts, inter-task communication, I/O and a few other things.

Exec Messages and Ports

Communication between tasks is accomplished through Exec objects called "messages" and "ports". A message is a block of memory that is used to send data from one task to another. The block of memory, containing a "message" data structure, belongs to the sending task. A message data structure is declared in the TDI package as:

```
Message = RECORD
mnNode      : Node;
mnReplyPort : MsgPortPtr;
mnLength    : CARDINAL;
END;
```

The mnNode field is used by Exec to manipulate messages. The mnLength field specifies the size of the message in bytes. The mnReplyPort field can be set up by the message's sender, if the sender wants to receive a reply (more about this in a while).

The sending task can pass ownership of that message (and, thereby, effectively send it) to the receiving task by calling the Exec procedure PutMsg. The receiving task can obtain the message by calling GetMsg. These calls are declared as:

```
PROCEDURE PutMsg(port : MsgPortPtr; message : MessagePtr);
    (* 'message' is a pointer to the message to be sent. *)
    (* 'port' is a pointer to the destination port. *)

PROCEDURE GetMsg(port : MsgPortPtr): MessagePtr;
(* 'port' points to the port from which to get the message. *)
(* GetMsg returns the address of the gotten message. *)
```

A "port" is the mailbox into which messages are deposited. It really is a data structure that has a name by which it is known to the sending task. As you can see in the PutMsg call, you must specify which port to put the message into. PutMsg appends the sent message to the end of the list of existing messages for that port. GetMsg removes a message from the head of that list. A port can be created using the CreatePort call, and it can be found through its name by using the FindPort call. These procedures are defined as:

```
PROCEDURE CreatePort(VAR name : ARRAY OF CHAR;
                     pri : INTEGER): MsgPortPtr;
(* 'name' - a null-termated string                          *)
(* 'pri' - priority to be assigned to the port (-127 to + 127) *)
(* CreatePort returns a pointer to the new port            *)

PROCEDURE FindPort
    (VAR name : ARRAY OF CHAR): MsgPortPtr;
    (* 'name' is the name of the port to find. *)
    (* FindPort returns the address of the port *)
```

Any task that needs to receive messages must have a port defined. A common practice is for the receiver to return the message to its rightful owner once it's done with it; after all, the message is a block of memory, and it isn't nice to steal someone else's memory. The receiving task can return the message by calling the ReplyMsg Exec routine, which is defined as:

```
PROCEDURE ReplyMsg(message : MessagePtr);
```

continued...

Notice that no port is specified in ReplyMsg. ReplyMsg will return the message to the port which is pointed to by the mnReplyPort field in the message. Therefore, any sending task that wants to have its message returned to it must have its own port, and must set the message's mnReplyPort field to point to the port.

Message passing is a very important aspect of the Amiga system, and is used extensively in the initiation of I/O activity and communication with devices.

Devices and I/O Requests

The ROM Kernel Manual defines a device as follows:

"A device in its purest sense is an abstraction that represents a set of well defined interactions with some form of physical media."

To put it another way, a device is a data structure which allows you to interface with some Input/Output unit (such as the serial port, the parallel port, the disk drive, or the console).

A device "unit" is an instance of the device, and while it shares the same data structure with all other units of the same device it operates independently of the other units. For example, each floppy drive in a two-drive system is a unit of the same device.

Each device on the Amiga has a name (a null-terminated string) which is used to identify it. For example, the serial device's name is simply "serial.device". A device is accessed by a user task through the OpenDevice call. Once a task is done using the device it should call CloseDevice. For every call to OpenDevice, there must be a matching call to CloseDevice. These calls are defined as:

```
PROCEDURE OpenDevice
        (VAR devName : ARRAY OF CHAR;
        unitNum : LONGCARD;
        ioRequest : ADDRESS;
        flags : LONGCARD): LONGCARD;

(* 'devName' is the name of the device                    *)
(* 'unitNum' is the number of the unit to be used.        *)
(* 'ioRequest' is the address of the IO Request block used *)
(*          to communicate with this device.              *)

PROCEDURE CloseDevice(ioRequest: ADDRESS);
```

A device receives I/O request through messages that contain various requests. These messages are actually instances of some data structure that contain – at the very least – the IORequest record, which is defined as follows:

```
IORequest = RECORD
ioMessage : Message;
ioDevice  : DevicePtr;
ioUnit    : UnitPtr;
ioCommand : CARDINAL;
ioFlags   : ioFlagSet;
ioError   : BYTE;
END;
```

The ioDevice and ioUnit fields are set up by the device itself when a call is made to OpenDevice. The ioCommand is the field used to specify the actual IO command requested, while ioFlags is used to specify special options. Any errors that may occur are returned in the ioError field, upon completion of the request. The ioMessage field is used by the device to return the I/O request upon completion to the originating task. Therefore, it is up to the originating task to set ioMessage properly before the I/O request is done (i.e. the mnReplyPort must be set to point to the task's reply port).

An I/O request block might contain more than an IORequest, though, all depending on the nature of the request. Standard I/O requests use a slightly augmented IORequest record which is called IOStdReq, and it looks like this:

```
IOStdReq = RECORD
ioReq     : IORequest;
ioActual  : LONGCARD;
ioLength  : LONGCARD;
ioData    : ADDRESS;
ioOffset  : LONGCARD;
END;
```

The ioLength field specifies how many bytes of data are to be transferred in the I/O request. The number of actual bytes transferred is returned in ioActual. If ioActual differs from ioLength, then an error must have occurred, and ioReq.ioError can be examined to determine the specific error. The ioData field points to the data itself.

The various devices may use different I/O request blocks, all of which consist of an IOStdReq record and some additional fields. The serial device, for example, requires a data structure called IOExtSer which is defined in the TDI package as:

```
IOExtSer = RECORD
ioSer       : IOStdReq;
ioCtlChar   : ARRAY (0..3) OF CHAR;
ioRBufLen   : LONGCARD;
ioWBufLen   : LONGCARD; (* is an error in the TDI *)
(* release. It should be ioExtFlags *)
ioBaud      : LONGCARD;
ioBrkTime   : LONGCARD;
ioTermArray : IOTArray;
ioReadLen   : BYTE;
ioWriteLen  : BYTE;
ioStopBits  : BYTE;
ioSerFlags  : SerFlagSet;
ioStatus    : SerStatusSet;
END;
```

More on the IOExtSer record and the serial device later on.

Communicating with Devices

Exec provides four functions for interfacing with devices. These general functions are independent of the actual device or the command requested. Rather, they deal with the I/O request block as a whole.

DoIO

The most commonly used I/O function. It sends an IO request to the device, and waits for the request to complete. This is called "synchronous" I/O, as the calling task is suspended until the I/O is complete. DoIO sends the IO request block to the device (through an implicit PutMsg to the devices port) and suspends the calling task until the device returns the IO request block to the task's reply port (i.e. does a ReplyMsg). DoIO does an implicit GetMsg, removing the IO request block from the calling task's reply port.

SendIO

This function sends an IO request to the device but does not wait for the request to complete. Rather, it returns immediately to the calling task, which is then free to do other things. This is called "asynchronous" I/O. It is up to the calling task to check for the completion of the I/O. Care must be taken not to modify the the IO request block while the device has ownership of it.

WaitIO

This function waits for the completion of a previously initiated asynchronous I/O request. The calling task is suspended until the I/O is complete. WaitIO performs an implicit GetMsg, removing the IO request block from the calling task's reply port.

continued...

CheckIO

A function to test whether an asynchronous I/O has completed. Upon completion of the IO, the IO request block will be placed in the task's reply port.

Standard IO Commands

There are eight standard commands to which all devices are supposed to respond. If a device does not support a specific command, it should return an error indicating that the command isn't supported.

CmdReset – resets the device unit, returning it to its default configuration and aborting all pending I/O. Any related hardware is reset too, and any internal data buffers associated with the device unit get cleared.

CmdRead – reads a specified number of bytes from a device unit into a data buffer pointed to by the ioData field. The number of bytes to be read is specified in the ioLength field, and the number of bytes actually read is returned in the ioActual field. If ioActual does not equal ioLength, an error must have been encountered, and ioError should contain the related error code.

CmdWrite – writes a specified number of bytes to a device, from a data buffer. The ioData field points to the area in memory into which the bytes are placed. The ioActual, ioLength and ioError fields are used in the same manner as with the CmdRead command.

CmdUpdate – forces all internal data buffers out to the physical device. Usually the device performs this operation implicitly, but this command causes the device to do so explicitly.

CmdClear – clears all internal data buffers without forcing the data to the physical device. The data is lost.

CmdStop – halts the device unit. Further I/O requests can still be queued, but the device won't service them.

CmdStart – restarts a stopped device unit.

CmdFlush – aborts all I/O requests. All requests are returned with an error indicating that they were aborted.

There are other commands which are device specific, and won't be covered here.

The Serial Device

The serial port can be opened in either exclusive access mode (meaning only on task can use it) or in shared more (allowing multiple tasks to do serial I/O), and it can be set to operate at different baud rates. Handshaking can also be specified. Handshaking means that the serial port has control of the flow of incoming data; it can tell the sender of the data (whoever is on the other side of the serial connection) to stop or resume transmitting it.

Handshaking and access mode must be specified before the device is opened (i.e. before the call to OpenDevice is made). Other parameters can be set afterwards using the SDCmdSetParams serial I/O command. Note, however, that a parameter change cannot occur while an I/O request is being processed.

Opening the Serial Device

The typical way of opening the serial device is as follows. First, create a reply port to which the serial device can return the IO request block. It can be quite convenient to have to ports, one for writing requests and one for reading. This way, both writing to the serial port and reading from it can be done simultaneously. An IO request block has to be allocated (again, it's quite handy to have on IO request

block for reads and one for writes). There is a function called CreateExtIO which allocates extended IO requests and sets them up to point to the given port. It is declared as:

```
PROCEDURE CreateExtIO(port : MsgPortPtr;
                      size : LONGCARD): IORequestPtr;
(* 'size' specifies the size of the extended IO request      *)
(*        block to be allocated.                             *)
(* CreateExtIO returns the address of the newly allocated    *)
(*        IO request block.                                  *)
```

For example:

```
VAR
  Buffer      : ARRAY (0..99) OF CHAR;
  ReadPort    : MsgPortPtr;
  ReadRequest : IOExtSerPtr;
  Result : LONGINT;
BEGIN
(* Create the read port and call it 'ReadMySerial' *)
  ReadPort := CreatePort("ReadMySerial",0);
(* create an IOExtSer structure by calling CreateExtIO *)
  ReadRequest := CreateExtIO(ReadPort,TSIZE(IOExtSer));
(* Prepare to open the device in shared mode without hand-
shaking *)
  ReadRequest^.ioSerFlags :=
SerFlagSet{SerShared,SerDisabled};
  IF OpenDevice(SerialName,0,ReadRequest,0)<>0 THEN
    (* Oops! We got a problem! Couldn't open device! *)
  END;
```

When the serial device is finally opened by a call to OpenDevice, it allocates an input buffer of the size last used (512 bytes is the minimum and the default). As with any of the other serial port parameters, the size of the input buffer can be changed by using the SDCmdSetParams command. The OpenDevice call initializes all the parameter fields of the IO request to the values last used.

Reading and Writing

Reading from the serial port is pretty simple. Set the ioData field to point to the address in memory into which you want to place the data. Set the ioLength field to the number of bytes you want to read, and set ioCommand to CmdRead. To continue the above example:

```
  WITH ReadRequest^ DO
    ioSer.ioReq.ioCommand := CmdRead;
    ioSer.ioData := ADR(Buffer);
    ioSer.ioLength := 100; (* read 100 bytes *)
  END;
  Result := DoIO(ReadRequest^.ioSer.ioReq);
  IF Result <> 0 THEN
(* Oops! We have an error! *)
  END;
```


If you use the above code (using DoIO), your task will be suspended (put to sleep) until the I/O is completed. Alternatively, you could use SendIO, and in this case, your task will continue running and therefore it could do other things while the I/O request is being processed. You can occasionally call CheckIO to see if the I/O has completed. A call to WaitIO will suspend your task until the asynchronous request completes.

Writing to the serial device is very much like reading from it. Set ioData to point to the data to be sent through the serial port. Set ioLength to the number of characters to be sent. Set ioCommand to CmdWrite, and you're all set! Now all you have to do is call DoIO or SendIO, depending on the type of I/O you want to do.

Serial Parameters

As mentioned before, all the serial parameters are initialized during the call to OpenDevice. You can modify them using the SDCmdSetParams command. These parameters are actually fields in the IOExtSer record (which was described earlier).

continued...

ioCtlChar – Control characters to use for XON, XOFF, INQ, ACK in handshaking. Currently only XON and XOFF are supported.

ioRBufLen – Size of the buffer that the serial device should allocate for incoming data. Minimum size is 512 bytes.

ioExtFlags – Reserved for future use. Incorrectly labeled in the TDI release as ioWBufLen.

ioBaud – The actual baud rate. Anywhere from 110 to 292000.

ioBrkTime – Specifies how long a break signal lasts (in microseconds).

ioTermArray – An array of eight bytes, specifying which characters will cause a read to terminate. You can read more about that in the RKM.

ioReadLen – Specifies how many bits per read characters. Usually 7 or 8.

ioWriteLen – Specifies how many bits per write characters. Usually 7 or 8. ioStopBits – Specifies how many stop bits to use. Usually 1 or 2.

ioSerFlags – Allows you to specify such things as handshaking, access mode, parity, and more.

ioStatus – Returns the status of the serial port.

About the program

The accompanying listings contain two modules: Serial and ModemDemo. I developed module Serial as a general serial port support module while working on a program to interface to a BSR X-10 PowerHouse Computer Interface unit. The ModemDemo module is a very simple (stress on simple) communication program. It was written just to demonstrate how easy it is to use the Serial module. ModemDemo also uses the MyRawInOut module that I presented in a previous article. There is one thing wrong with this program: it contains a "busy loop", that is it runs continuously even if there is no I/O from the serial device or the console. This is a no-no on a multitasking operating system, as it just wastes CPU resource. In a future article I'll show how to modify the code to turn this program into a "well-behaved" one.

Further Reading

This article covered the topics of I/O and devices only in a broad fashion. More detailed information can be found in the ROM Kernel Manual (Volume 1). Also recommended, is the book "Programmer's Guide to the Amiga" by Robert Peck (Sybex).

```
DEFINITION MODULE Serial;

        (* * * * * * * * * * * * * * * * * * * * * * * * * * * *)
        (*                                                       *)
        (*  General Serial Port Support Module.                  *)
        (*  (c) Copyright 1986 by Steve Faiwiszewski.  *)
        (*                                                       *)
        (*  This program may be freely distributed       *)
        (*  for non-commercial use only.   It may not    *)
        (*  be sold.                                     *)
        (*                                                       *)
        (*  Please leave this notice intact.                     *)
        (*                                                       *)
        (* * * * * * * * * * * * * * * * * * * * * * * * * * * *)

FROM SYSTEM IMPORT BYTE;

PROCEDURE OpenSer(BaudRate : LONGCARD;
                  StopBits, BLength : BYTE;
                  XON, XOFF, IND, ACK : CHAR);

PROCEDURE CloseSer;

PROCEDURE SerWrite(VAR Buffer : ARRAY OF BYTE; Length: LONGCARD);

PROCEDURE SerRead(VAR Buffer : ARRAY OF BYTE; Length : LONGCARD);

PROCEDURE QueueSerRead(VAR Buffer: ARRAY OF BYTE);
(* Do a Read for 1 character asynchrously.  That is, do a BeginIO
*)
(* instead of a DoIO
*)

PROCEDURE ProcessSerMessage(VAR Buffer : ARRAY OF BYTE; Length :
LONGCARD);
(* Wait for a BeginIO to finish *)

PROCEDURE QuerySer() : LONGCARD;
(* Find out how many characters are waiting to be read *)

END Serial.
```

```
IMPLEMENTATION MODULE Serial;

        (* * * * * * * * * * * * * * * * * * * * * * * * * * * *)
        (*                                                       *)
        (*  General Serial Port Support Module.                  *)
        (*  (c) Copyright 1986 by Steve Faiwiszewski.  *)
        (*                                                       *)
        (*  This program may be freely distributed       *)
        (*  for non-commercial use only.   It may not    *)
        (*  be sold.                                             *)
        (*                                                       *)
        (*  Please leave this notice intact.                     *)
        (*                                                       *)
        (* * * * * * * * * * * * * * * * * * * * * * * * * * * *)

FROM SerialDevice IMPORT SerialName, SDCmdSetParams, SDCmdQuery,
IOExtSer,
                         SerFlagSet, SerShared, SerDisabled;
FROM Ports              IMPORT MsgPortPtr;
FROM PortUtils          IMPORT CreatePort, DeletePort, CreateExtIO,
DeleteExtIO;
FROM IO                 IMPORT IORequestPtr,DoIO, SendIO, WaitIO,
CmdWrite,
                         CmdRead;
FROM Devices            IMPORT OpenDevice, CloseDevice;
FROM InOut              IMPORT WriteString, WriteLn;
FROM SYSTEM             IMPORT BYTE, NULL, ADR, TSIZE;

TYPE
    IOExtSerPtr = POINTER TO IOExtSer;

VAR
    ReadPort, WritePort : MsgPortPtr;
    ReadRequest, WriteRequest : IOExtSerPtr;

PROCEDURE Cleanup(n: CARDINAL);
BEGIN
```

```
   IF n >= 5 THEN
      DeleteExtIO(IORequestPtr(WriteRequest),TSIZE(IOExtSer));
   END;
   IF n >= 4 THEN
      DeletePort(WritePort)
   END;
   IF n >= 3 THEN
      CloseDevice(ReadRequest)
   END;
   IF n >= 2 THEN
      DeleteExtIO(IORequestPtr(ReadRequest),TSIZE(IOExtSer))
   END;
   IF n >= 1 THEN
      DeletePort(ReadPort)
   END;
END Cleanup;

PROCEDURE Abort(VAR msg : ARRAY OF CHAR; n : CARDINAL);
BEGIN
    WriteLn;
    WriteString(msg); WriteLn;
     IF n > 0 THEN Cleanup(n) END;
    HALT
END Abort;

PROCEDURE OpenSer(BaudRate : LONGCARD;
                  StopBits, BLength : BYTE;
                  XON, XOFF, IND, ACK : CHAR);
VAR
   Result : LONGINT;
BEGIN
       (* Create the read port and call it 'ReadMySerial' *)
    ReadPort := CreatePort("ReadMySerial",0);
     IF ReadPort = NULL THEN Abort('Could not create ReadPort!',0)
END;

    ReadRequest := CreateExtIO(ReadPort,TSIZE(IOExtSer));
    IF ReadRequest = NULL THEN
       Abort('Could not create ReadRequest!',1)
    END;

    ReadRequest^.ioSerFlags := SerFlagSet{SerShared,SerDisabled};

    IF OpenDevice(SerialName,0,ReadRequest,0)<>0 THEN
        Abort('Could not open Serial device for read!',2)
    END;

    WritePort := CreatePort("WriteMySerial",0);
     IF WritePort = NULL THEN Abort('Could not create WritePort!',3)
END;

    WriteRequest := CreateExtIO(WritePort,TSIZE(IOExtSer));
    IF WriteRequest = NULL THEN
       Abort('Could not create WriteRequest!',4)
    END;

(* now, since ReadRequest is all set up, but WriteRequest isn't,
*)
(* simply copy WriteRequest from ReadRequest, but point
*)
(* WriteRequest's reply port to its own port.
*)

    WriteRequest^ := ReadRequest^;
    WriteRequest^.ioSer.ioReq.ioMessage.mnReplyPort := WritePort;

    WITH ReadRequest^ DO
        ioSerFlags := SerFlagSet{SerShared,SerDisabled};
        ioBaud := BaudRate;
        ioStopBits := StopBits;
        ioReadLen := BLength;
        ioWriteLen := BLength;
        ioCtlChar[0] := XON;
        ioCtlChar[1] := XOFF;
        ioCtlChar[2] := IND;
        ioCtlChar[3] := ACK;
        ioSer.ioReq.ioCommand := SDCmdSetParams;
    END;
    Result := DoIO(ReadRequest^.ioSer.ioReq);
     IF Result <> 0 THEN
        Abort('** Error during OpenSer: Could not change parameters
**',6)
    END;
END OpenSer;

PROCEDURE CloseSer;
BEGIN
    Cleanup(99);
END CloseSer;

PROCEDURE SerWrite(VAR Buffer : ARRAY OF BYTE; Length: LONGCARD);
(* Write 'Length' number of bytes from 'Buffer' to the serial port
*)
VAR
   Result : LONGINT;
BEGIN
    WITH WriteRequest^ DO
        ioSer.ioReq.ioCommand := CmdWrite;
        ioSer.ioData := ADR(Buffer);
        ioSer.ioLength := Length;
    END;
(*     Result := DoIO(IORequestPtr(WriteRequest)); *)
    Result := DoIO(WriteRequest^.ioSer.ioReq);
     IF Result <> 0 THEN
        WriteString('** Error during SerWrite **'); WriteLn
    END;
END SerWrite;

PROCEDURE SerRead(VAR Buffer : ARRAY OF BYTE; Length : LONGCARD);
(* Read 'Length' number of bytes from 'Buffer' from the serial
port *)
VAR
   Result : LONGINT;
BEGIN
    WITH ReadRequest^ DO
        ioSer.ioReq.ioCommand := CmdRead;
        ioSer.ioData := ADR(Buffer);
        ioSer.ioLength := Length;
    END;
    Result := DoIO(ReadRequest^.ioSer.ioReq);
     IF Result <> 0 THEN
        WriteString('** Error during SerRead **'); WriteLn
    END;
END SerRead;

PROCEDURE QueueSerRead(VAR Buffer: ARRAY OF BYTE);
(* Queue up (asynchrously) a request to read one byte into
'Buffer' *)
BEGIN
    WITH ReadRequest^ DO
        ioSer.ioReq.ioCommand := CmdRead;
        ioSer.ioData := ADR(Buffer);
        ioSer.ioLength := 1;
    END;
    SendIO(ReadRequest^.ioSer.ioReq);
END QueueSerRead;

PROCEDURE ProcessSerMessage(VAR Buffer : ARRAY OF BYTE; Length :
LONGCARD);
(* Wait for the asynchronous read request to complete, and read
'Length' *)
(* bytes into 'Buffer'
*)
VAR
    Result : LONGINT;
BEGIN
    Result := WaitIO(ReadRequest^.ioSer.ioReq);
     IF Result = 0 THEN
       SerRead(Buffer,Length)
    ELSE
        WriteString('** Error during WaitIO in ProcessSerMessage **');
       WriteLn
    END;
END ProcessSerMessage;

PROCEDURE QuerySer() : LONGCARD;
(* Returns the number of characters that are waiting to be read
from the *)
(* serial port.
*)
VAR
   Result : LONGINT;
BEGIN
    WITH ReadRequest^ DO
       ioSer.ioReq.ioCommand := SDCmdQuery;
       Result := DoIO(ReadRequest^.ioSer.ioReq);
        IF Result <> 0 THEN
           WriteString('** Error during SerRead **'); WriteLn
       END;
       RETURN(ioSer.ioActual)
   END;
END QuerySer;

END Serial.
```

•AC•

continued from page 73

Once you've finished fooling with parameters, the connection prep is nearly complete. Now you need only establish whether you will be calling or receiving. The TeleGames manual suggests a vocal connection to get things settled. You can set your status via the "Telecommunications Mode" option under the "Telecomm" menu.

The final step on behalf of the "sender" is to activate the "Dial Telephone Number" option under the "Telecomm" menu to accomplish the obvious. Once the receiving computer answers, the teleconnection is established and players can type messages back and forth and start a new game or resume a saved game. Message lines along the bottom of the game screen give you plenty of "talking" room. The "Telecomm" menu allows you to "Transmit Game and Play" or "Disconnect" (if your opponent is already getting on your nerves!).

TeleGames . . . Documentation

The forty page TeleGames manual provides a concise overview of program use and some nice "extras." The manual is, for the most part, clearly written and free of unnecessary jargon.

Some of the "extras" come in the form of useful appendices. Appendices A and B outline use of TeleGames from the RamDisk or a Hard Disk. Appendix C talks about the programs' two main directories and how they can be used to your advantage. The final appendix is extremely helpful and would make a great addition to many manuals — a trouble shooting guide. A list of possible solutions are given for five all-too-realistic "boo-boos" you just might make.

If you are still puzzled about something, even after reading the manual, TeleGames provides a bulletin board service. Posting your problem here will get you an answer "within 24 hours," according to the manual. Mailing in the registration card entitles you to BBS access, as well as free updates and revisions. One of the keys to a successful product is conscientious support. TeleGames has this area well covered.

You may follow the telecommunications instructions "to a T" and still find yourself all alone and unconnected. Before re-packing your game for return, though, consult the manual under the "Null Modem Connect." If your modem is not "AT" standard, you may find a solution through some outside telecommunications software. The manual outlines use of such software with the "Null" option.

TeleGames . . . TeleGripes

Overall, I would give TeleGames high marks, but I do have a few minor grumbles.

The first deals with the verification process. I support such protection wholeheartedly, but I would also like to have more than one chance before the program defaults all the way back to the Workbench. It is just too easy to hit the wrong key or look up the wrong word. A second chance at verification would allow for human error while maintaining the desired protection.

My second gripe comes from personal experience. What do you do if there are no opponents available, either at home or by modem? Play against the computer, right? Wrong. TeleGames does not provide play versus the computer. A computer opponent would make a nice addition for those lonely times.

My final problem with TeleGames is one that is a little hard to see . . . literally. The light type in the manual, despite its copyright purpose, is difficult to read. This problem is a minor one, though, and can be alleviated with a bright bulb.

TeleGames . . . Overview

TeleGames deserves all the praise given to it, here and elsewhere. The few problems mentioned above are only minor nuisances. The concept is revolutionary, the graphics are great and gameplay is consistent. The teleconnection process is as simple as you could ever hope for. If you have an interest in telecommunications, or if your fiercest competitor is a long distance call away, TeleGames is a program you should not pass up.

•AC•

68000 Assembly Language Programming

Display Routines

by Chris Martin

Last month I presented a program (that you typed and compiled, I hope) which opened a simple Intuition graphics screen. This month and next month, I'll discuss the graphics process in as much depth as possible. Remember that the internal graphics routines can be divided into two separate types: drawing routines and display routines. Here we will discuss the display routines (I know, you can't wait to draw pictures in assembly language, but we must take this one step at a time!)

Raster Displays

There are two main types of computer display: vector display and raster display. A vector display is completely made with lines, as in arcade games such as Star Wars and Asteroids. Some sophisticated graphics terminals have this type of display because of it's high resolution – the graphics "jaggies" are nonexistent in vector displays. Most home computers, including the Amiga, have a raster display. A raster display is made by a beam of electrons, called a raster beam, that sweeps across all rows of the display screen, drawing dots as it goes.

The Amiga can have a resolution of 320 columns by 200 rows. Thus, in this display mode the raster beam sweeps across 200 rows, all done in under 1/60 of a second. The Amiga can have other resolutions with 400 rows, for example 640 by 400. In this mode the display acts a little bit differently than described above. Instead of sweeping across all 400 rows one after another, the raster beam is instructed by the computer to draw only the odd rows (1,3,5, ... 399) in 1/60 of a second, then go back to the top of the screen and draw even rows (2,4,6, ... 400) in the next 1/60 of a second. This mode is called interlace mode, and because the computer must draw the screen twice, there is a noticable amount of flicker with certain color combinations, and because of the time needed to redraw the entire screen (2/60 of a second) the speed of the computer is somewhat slower.

Colors

Everything would be much easier to explain if we could only work in two colors, but because colors are nice to have around, I think I'll explain how they tie into the display!

The Amiga graphics system has 32 color registers, each having a number 0 to 31 and having a description of the red, green, and blue content of the color. When describing he attributes of a display screen, one must also tell the system how many "bit planes" are to be used. Each bit plane is the size of your screen, and they are in effect overlapped and combined to create colors. The number of colors relates to the number of bit planes used (remember, the more bit planes, the more memory needed). For example in a one bit plane display, there is only one drawing color available and one background color. Thus, if a bit is 1, color register one would be accessed to display a dot in that color. On the other hand if the bit read 0 the background color register zero would be displayed.

If more than one bit plane are used, the bits are combined to come up with a number corresponding to a color register. Because the Amiga can only display 32 colors at a time (except in "hold and modify" mode, which will be explained at a later date), a maximum of 5 bit planes may be used.

```
                number of bit planes
(NUMBER OF BIT COMBINATIONS = 2)   = NUMBER OF COLORS
```

In a 5 bit plane display, the number of colors would be

$$(2)^{5} = 32.$$

Here is an example of the bit combination process in the color selection process with five bit planes:

```
                          Bit 1 or 0   Combination    Color #

Plane 1 ---->  1   \
Plane 2 ---->  0   |
Plane 3 ---->  1   |--> 1 0 1 1 0   = 2+4+16 = 22
Plane 4 ---->  1   |
Plane 5 ---->  0   /   ( 16, 4, 2 )
```

Graphics Structures

Here I will discuss and define the various graphics structures that you use to create a display. Please note that this information is very "low level" and many Inutition functions do most of this parameter definition for you. The entire display is produced from the paramenters in the VIEW

continued...

Index of Advertisers

A-Squared Distributions	37
Absoft	27
Aegis Development	35
Ami Expo	53
Aminetics	110
Applied Visions	CII
BCD	59
Byte by Byte	CIV
Central Coast Software	51
Comp-U-Save	90
Computer Visual Services	26
Computerware	87
Creative Solutions	88
D-Five Associates	69
Data Solutions	92
Discovery Software	64,65
Eagle Tree Software	85
Fuller Computer Systems	47
Gimpel Software	18
Hash Enterprises	42
Hilton Android	86
Interactive MicroSystems	115
InterActive Softworks	91
Kent Engineering & Design	96
Kline-Tronics	84
Lattice	7
Lightning Publishing	54
Lynn's Luna C	24
Manx C Software	57
Megatronics	13
Michigan Software	99
Microbotics	11
Microillusions	CIII
Mystic Plain Software	81
NewTek	1
Newwave Software	106
OTG Software	70
Pacific Peripherals	28
Peacock Systems	82
Pheonix Electronics	102
PiM Publications	120, 121 122, 128
Proloific Inc.	3
Prospect Software	97
PVS Publishing	16
ReadySoft Inc.	25
Rittinghouse Software Development Co.	68
Second Source Systems	6
Software Supermarket	94
Software Terminal	113
Speech Systems	74
SunRize Industries	77
Syndesis	95
T.S.R. Hutchinson Co.	80
TDI Software	105
The Memory Location	98
The Other Guys	17,39
The Right Answers Group	63
TRU-IMAGE	3
Westcom Industries	107

structure. A VIEW consists of one or more VIEWPORTS. VIEWPORTS define sections of the VIEW and may have different resolutions and color sets. When an Intuition screen is created, the system sets up a screen with a VIEW structure of its own. When you drag the screen down using the menu bar, you are moving the VIEW structure down to reveal the VIEW underneath. The VIEW structure has three major parameters: ViewPort, DxOffset, and DyOffset. ViewPort points to the first VIEWPORT structure in a linked list. If more than one VIEWPORT is to be used, the first VIEWPORT structure will point to the second, the second to the third, and so on. The parameters DxOffset and DyOffset control the position of the VIEW on the display. These indicate the coordinated of the upper left corner of the VIEW.

Structure: **VIEW** Include File: **Graphics/View.I**

Components:

LONG	v_ViewPort	Pointer to the first ViewPort.
LONG	v_LOFCprList	Display Lists (internally controlled).
LONG	v_SHFCprList	
WORD	v_DxOffset	X & Y coordinate offsets for View.
WORD	v_DyOffset	
WORD	v_Modes	Display modes for entire View (see below).

Display Modes:

V_DUALPF	Dual Playfield Mode
V_HIRES	640 columns across screen
V_LACE	Interlace mode, 400 rows
V_HAM	Hold and Modify Mode
V_SPRITES	Include sprites on screen

NOTE: The display modes can be combined by separating the parameters by a "!" character. E.G. V_HIRES!V_LACE creates 640x400 screen.

The VIEWPORT structure is the heart of the Amiga display system. Each VIEWPORT can have its own colors, resolutions, and display memory. ViewPorts must be separated by one or more rows, and may not overlap. The vp_Modes parameter works in the same way that the View v_Modes parameter works, except it only affects the portion of the display defined by the ViewPort structure.

Structure: **VIEWPORT** Include File: **Graphics/View.I**

Components:

LONG	vp_Next	Next ViewPort in a linked list
LONG	vp_ColorMap	Points to a ColorMap structure
LONG	vp_DspIns	These four are for internal use
LONG	vp_SprIns	
LONG	vp_ClrIns	
LONG	vp_UCopIns	
WORD	vp_DWidth	Width of this ViewPort
WORD	vp_DHeight	Height of this ViewPort
WORD	vp_DxOffset	The X & Y coordinates for screen location
WORD	vp_DyOffset	
WORD	vp_Modes	Display modes for this individual ViewPort (see above)
LONG	vp_RasInfo	Pointer to RasInfo structure

The RasInfo structure relates to the position of the graphics to be displayed the ViewPort. In the structure is a pointer to the graphics, stored in a BitMap. Also there are parameters that define the coordinate position of the upper left of the bitmap in relation to the upper left of the ViewPort.

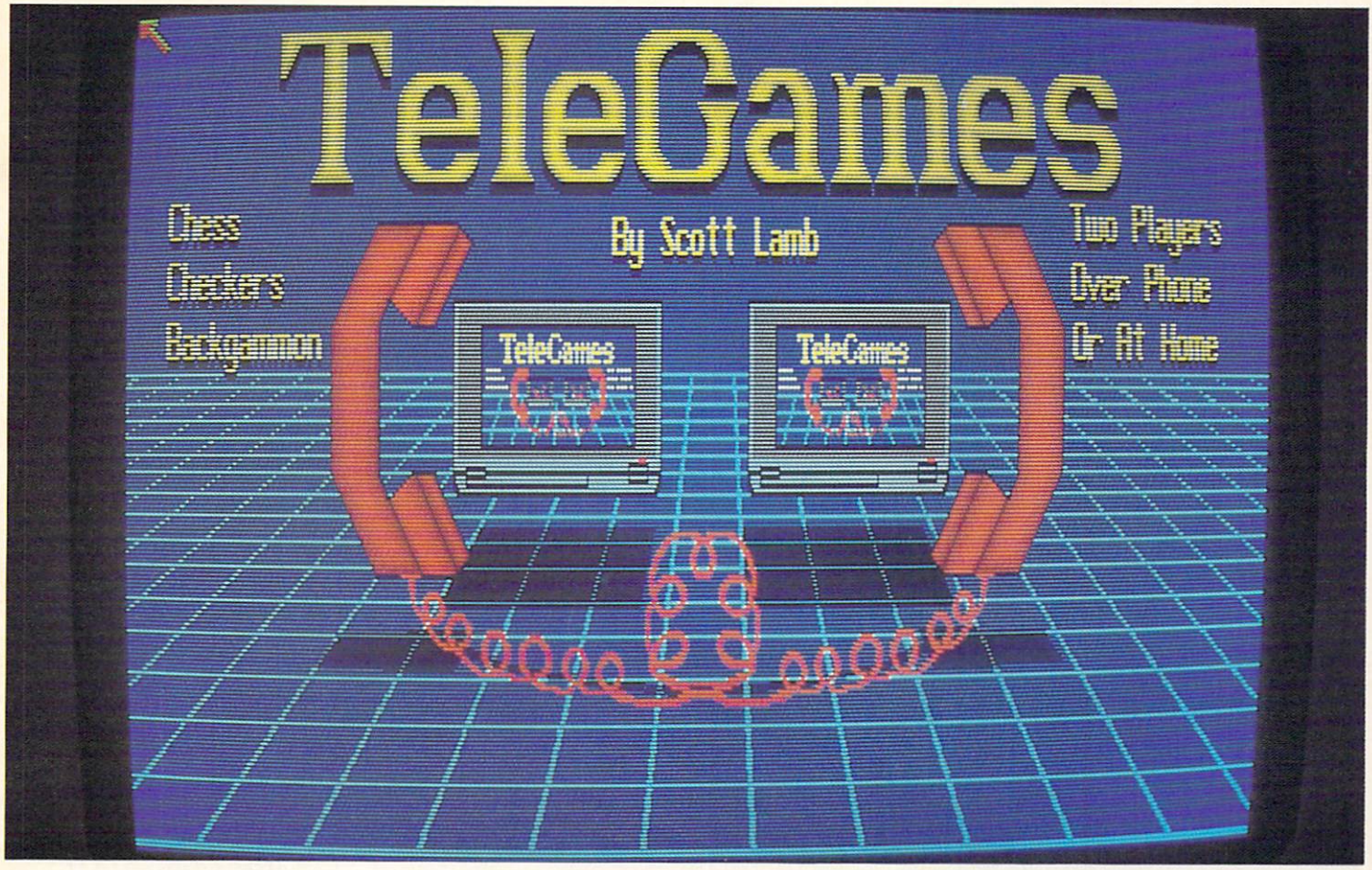
Structure: **RASINFO** Include File: **Graphics/View.I**

Components:

APTR (pointer)	ri_Next	Pointer to the next RasInfo
LONG	ri_BitMap	Pointer to a BitMap structure
WORD	ri_RxOffset	X & Y coordinates of the BitMap to be displayed in the upper left of the corresponding ViewPort.
WORD	ri_RyOffset	

The BitMap strcture points to the actual graphic bit planes, and holds the size and depth of your graphic image.

continued on page 113



TeleGames is what you've waited for. The Future is here.

TeleGames allows you to use your computer and modem to play Chess, Checkers and Backgammon with a human opponent over the telephone. Only $34.95!

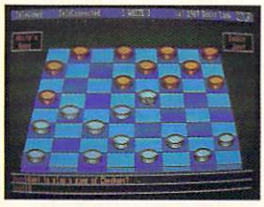
TeleGames Features

* Chess * Checkers * Backgammon
* Superb Graphic Game Simulations
* Smooth Depth Arranged Movement
* 4 angle 3D & 2D view perspectives
* Digitized Sound Effects
* Compatible with any modem
* 300, 1200, 2400, 9600 Baud
* Call originate or answer
* Null Modem Connect option
* Save Game & Transmit Game options
* Opponent File Directories
* Send and Receive Typed Messages
* Easy to Use Menus & Requesters
* All Official Game Rules Supported
* Play Over the Phone or at Home
* Legal Moves Graphically enacted on the TeleConnected computer
* Fully copyable to hard disks
* Upgrades available on our BBS

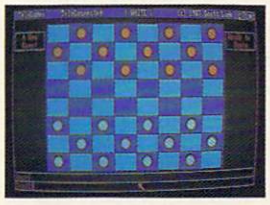
3D Chess

2D Chess

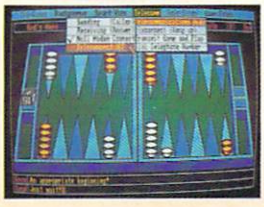
3D Checkers



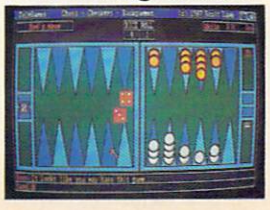
2D Checkers



3D Backgammon



2D Backgammon



If you Enjoy Telecomputing, You'll Love TeleGames!

Published by Software Terminal
3014 Alta Mere, Fort Worth, TX 76116
817-244-4150 Modem: 817-244-4151
Dealer Inquiries Invited

continued from page 112

Structure: **BITMAP** Include File: **Graphics/Gfx.I**

Components:

```
WORD    bm_BytesPerRow  The # of dots across divided by 8.
WORD    bm_Rows         The number of rows in the bitmap
BYTE    bm_Flags        Internal use flags
BYTE    bm_Depth        The number of bit planes (colors).
WORD    bm_Pad   Extra WORD of space used for "padding".
WORD    bm_Planes,1     Actual bit plane pointers.
WORD    bm_Planes,2     Notice there is room for up to 8 bit
WORD    bm_Planes,3     planes, although there is a
                        maximum of six per screen).
.
WORD    bm_Planes,8
```

Before creating a display, one must define the elements of these structures, then pass these structures to the ROM Kernal routines that builds the graphic picture.

Defining Structures

First of all, the computer must know where the skeleton of each structure is on the disk. You will notice that after each structure name I have listed the name of the include file and its directory on your assembly workdisk. At the very beginning of your program, you must INCLUDE these files as in the below example:

```
(3 tabs)        (3 tabs)
INCLUDE         "graphics/gfx.i"
```

For the first include files always include the following:

```
INCLUDE         "exec/types.i"
INCLUDE         "exec/funcdef.i"
```

The file "types.i" defines the length of the WORD, LONG, and BYTE identifiers. "Funcdef.i" creates macros (one command performs many) for calling ROM Kernal functions and passing parameters.

After the computer knows what the structure looks like, you need to fill in the elements. This is not a direct process, and may seem a bit confusing at first, but it all fits together. You must define two SECTIONs towards the end of your program. The first section will contain the names of your structures, the second will be the reserved space for the elements of the stuctures.

A SECTION can be defined in your source code in the following manner. First, give the SECTION a label. Next describe the attributes of the SECTIONS. For example,

```
  LABEL          NAME, TYPE
-------------------------------
 Uninitdata   section vars,bss
startbss
.
. (data)
.
endbss
```

This section's name is "vars" under the label of "Uninitdata" of type "bss". Overall, there are three types: CODE, DATA, and BSS. CODE, the default type defines a section of relocatable code. The assembler is automatically put into the CODE format from the start of your program. The type DATA consists of "initialized data", that is, cleared data space that is to be used by structure parameters. The final type, BSS, is uninitialized data space, used for pointers and miscellaneous data storage not grouped in any particular order. You define the start and end of the SECTION by using

```
STARTtype       e.g.    startbss

and

ENDtype         e.g.    enddata.
```

First, define a SECTION of BSS type containing LONG word (32 bits) space for what will be the pointers to your structures. For example:

```
Uninitdata    section  names,bss

startbss

myview        ds.l     1          * reserve space for 1 LONG (32 bits)
mybitmap      ds.l     1

endbss
```

Next, define the values of the elements of the structures to be used in a DATA type SECTION. The length of the space needed for each element is listed to the left of its name in my listing of the structures. For example:

```
Initdata      section  structures,data

startdata

viewport      dc.l     0          * no next ViewPort
              dc.l     0          * no ColorMap structure
              dc.l     0          * DspIns
              dc.l     0          * SprIns
              dc.l     0          * ClrIns
              dc.l     0          * UCopIns
              dc.w     320        * ViewPort width
              dc.w     100        * ViewPort height
              dc.w     1          * DxOffset
              dc.w     20         * DyOffset
              dc.w     HIRES      * Modes - 640 across, 200 down
              dc.l     0          * No RasInfo structure

enddata
```

Instead of using "ds", here we use "dc"; we are defining a constant, not space. Now that the structures are defined and their elements are filled, the structures can be passed to the routines that manipulate them.

Next month we'll use these structures and create our own non-intuition display, and I'll begin on a discussion of the drawing routines.

•AC•

Continued from page 43

To Reason
Starting Reason Version 2.02 requires the user to preform a warm boot (Ctrl Amiga™), since the entire 512K on a "standard" Amiga™ is neccessary to run the program. Mr. Nielsen has reported that an upgrade will be available by mid-November which will allow Reason to run in a standard Amiga enviornment on machines of 1 meg or more.

Reason's startup screen is the highest level menu in a completely menu driven program. From this point, the user can select the document and all options for proofreading. As each main selection is clicked, a smaller submenu is produced to access the desired criteria.

Since there is no access through the normal CLI or Intuition WorkBench, Reason has an option menu for selecting your input document by disk, directory and file.

One problem is, the document must have been saved in an ASCII or text format. Any special layout structure, characters, or typefaces would not be carried through, and the user is forced to make the required changes by hand in the original document under the original word processor program.

Output Mode/File is available through the next menu. Output may be directed to screen, disk, or printer.

The "Critique Document For Prose" selection allows the user to critique their text structure to *Instructional Text, Technical Memoranda,* or to *Use Custom Standards.* Instructional Text compares the document to Bell Labs training documents. Technical Memoranda compares the document to Bell Labs Technical Memoranda "judged good by document heads in the research area," per Reason's manual. Custom Standards allow the user to compare the prose to a standard established earlier under the *Extra* menu.

The user's next choice is *Critique Document for Style.* This menu accesses a variety of options to print sentences containing Passive Verbs, begining with an Expletive, and containing Noun Nominalizations and All (which shows the length and ARI: Automated Readability Index). There are also options for printing sentences longer or shorter than a certain number of words, with ARI greater or equal to a selected number of years schooling, or print statistics only.

Under the Proofread Document selection, the user can check for all or any of the following: spelling, double words, punctuation, diction, split infinitives. While in the Word Analysis selection, the user may choose only one available option: general diction, sexist terms, forms of "to be," acronyms, abstract words (detailed or summary) or a custom dictionary scan.

The General Structure menu allows the user to analyze and report on the general organization, general topics, sentence breakdown or syllable breakdown of the document. The Extra menu accesses the preferences screen and the input area where a custom prose standard file is built.

Summation
Reason™ is a vast and intense program which allows the micro computer user options never before available in proofreading a document. This preview in AC's word processing issue is included to introduce the user to the power that appears available through this software. However, this is only a preview. AC™ retains the right to exercise this program more completely. A more intense and complete review will be available in a future issue of Amazing Computing™.

•AC•

The AMICUS Network™

Desktop publishing

by John Foust

The Seybold Desktop Publishing Conference was held in Santa Clara California in mid-September. As you might imagine, the event was dominated by the Macintosh, followed closely by the IBM personal computer. Still, Commodore had a booth at the conference. With the latest desktop publishing programs, the colorful Amiga can offer an alternative to the grey world of the Mac and IBM.

At this show, I decided to be evangelical about the Amiga. I approached people from the aisles around the booth and asked them their impressions of what they saw. HAM pictures and paint programs astounded them. When I told them the price of a complete Amiga system with a monitor, extra disk drive and memory, they were taken aback – after all, a system that can do this should cost so much more – or so it seemed to them.

One side of the booth was devoted to a network of Amiga 2000s. At the end of the booth stood two expensive and high-quality laser printers. Next to this was Gold Disk, who used the printers to demonstrate features of their new desktop publishing program, Professional Page.

Also on the network was Word Perfect, demonstrating the Amiga version of their popular word processor. As you could imagine, this side of the booth demonstrated the way Amiga programs work together. At the opposite corner of the booth was New Tek, demonstrating their Digi-View digitizer. Professional Page integrated Digi-View digitized photographs and Word Perfect text into documents that were output to the high-end laser printers.

On the other side of the booth, Brown-Wagh showed Publisher 1000 and Microsearch showed City Desk. At least at Commodore's show appearances, the triumvirate of Amiga desktop software is Publisher, City Desk and Professional Page. Unfortunately, Professional Page has not yet shipped. Publisher 1000 and City Desk have been shipping for quite some time. Gold Disk was the first company with publishing software, when they shipped PageSetter last January. Professional Page is their latest entry in the leapfrogging battle of features and options in desktop programs.

There is an interesting trend taking place in Amiga page composition software. The low-end word processors are adding the ability to integrate graphics and color in documents, narrowing the gap between the low-end desktop software and the high-end word processor. New Horizons has ProWrite 2.0 with similar features and Progressive Peripherals has VizaWrite. At the show, Infinity Software demonstrated an early version of Shakespeare, another word processor with graphics and color.

Laser concepts

I encountered a common question from attendees. They wanted to know the Amiga screen image could be printed. After answering this question several times, I realized these people weren't asking the right question. They wanted to know if they could use these images in laser-printed documents. The answer is a qualified "yes." The answer touches on several concepts relevant to anyone who does graphic output on the Amiga.

The misunderstanding has to do with resolution. Any Amiga screen as a certain resolution, the number of separately discernable dots across the screen. Low resolution has 320 by 200 pixels (dots), high resolution has 640 by 400 dots. An Amiga video digitizer creates images that have the resolution of the screen.

In itself, video digitizer technology was unfamiliar to these Mac and IBM users. They are more familiar with page scanners. While the concepts are similar, the implementation and results are not the same, and this confused many observers.

A page scanner digitizes a printed page at very high resolutions. Ideally, a scanner system digitizes images at the same resolution at which images are output. In effect, a copying machine is a good page scanner. Many new copying machines are little more than scanners combined with laser printer engines.

Desktop publishers need a much higher resolution than the Amiga screen. They are most concerned with creating fine, sharp lines on a laser printer. Most laser printers control 300 dots per inch (DPI) to make images. The dots are a fixed size and darkness. There is no way to print a grey dot. On a laser-printed page, the entire

image is composed of dots that are either printed or not printed. To get shades of grey, alternate dots are used.

The resolution of output devices has always been much higher than the resolution of display screens. The resolution of a dot-matrix printer is about 72 DPI. Some 24-pin printers have about about 200 DPI, nearing the resolution of laser printers.

How would a laser printer print an Amiga screen? Each pixel on the screen is assigned a shade of grey. Because the number of dots per inch is much higher on the laser printer, each Amiga screen dot must be printed as a series of laser printer dots. This works out well for printing the shades of grey, because a certain number of dots are needed to represent different shades of grey. Because the dots are so small, the eye blends the dots and the lack of dots into a shade of grey.

So each Amiga pixel is printed as a block of laser printer dots. A grey Amiga screen pixel prints as a very small box filled with a pattern of laser printer dots to make that shade of grey. This same principle is used to print Amiga screens on dot-matrix printers.

If the Amiga screen is high resolution, it has 640 pixels across and 400 pixels high. If the box is only four laser dots square, for a total of sixteen dots, then this image would print about 8 1/2 inches wide, the width of a regular sheet of paper, and about 5 1/3 inches tall. Sixteen shades of grey are possible in each box.

How can a black-and-white laser printer make color output? By modeling the process used in printing ordinary color photographs. In conventional color printing processes, a color image is separated into its component colors. This is done by photographing the image with different color filters. Then the image is printed four times, once each with black, cyan (blue), yellow and magenta inks. The computer can perform the same separation of colors, producing four images that can be printed separately on the laser printer.

Accompanying this column are examples of a four-color color separation from Professional Page. This was a 320 by 400 pixel HAM image digitized with Digi-View. It was printed on a 1200 DPI Linotronic Imagsetter. This color separation feature is quite rare in microcomputer desktop publishing software. It made the Amiga stand out at the Seybold conference.

What is the answer to the question "How can the Amiga screen be printed?" The answer is, any way you want – on a black-and-white or color dot-matrix printer, on an ink-jet printer, or on a laser printer. No matter the output medium, a screen dump is just a screen dump. It will always look coarse because the screen is much lower resolution than the printer.

This can change. The Amiga can create graphics without the coarseness of the screen resolution. If the program uses a structured graphics standard such as the PostScript laser printer standard, it can make graphics as smooth as the output device can handle. Very few Amiga programs use this approach to graphics.

Amiga presence

I am sure many people were surprised to see Commodore at a show such as the Seybold Show. As I said, the show was dominated by the Macintosh and IBM. If the show exhibits were broken down by percentages, about ninety percent was devoted to the Macintosh and nine percent for the IBM AT clones. Commodore fit into the remaining one percent. I did see one or two Atari STs at the show, tucked into a small booth on the other side of the show.

The Macintosh software for manipulating laser-printed images is astounding. They have the standard PostScript language for creating graphics on laser printers. There are entire classes of software available for the Macintosh that do not exist on the Amiga, such as structured drawing programs like Adobe Illustrator.

The Commodore booth was popular among attendees, though. The flash and the graphics of the Amiga drew crowds and held them momentarily. Anyone at the Seybold conference was there for desktop publishing, and the Amiga couldn't hold their attention strictly on that criteria. You could use one hand to count the programs that might come close to what a Macintosh can do with a laser printer. The demonstrations may have been attractive, but because the Amiga wasn't Macintosh compatible, most people viewed it as a curiousity.

Commodore spent a lot of money to come to this show. It is nice to see that they are pinpointing specific market segments such as desktop publishing, but their efforts may be misdirected. In this case, it is hard to justify an entire booth to showcase less than six products that have any remote connection to desktop publishing.

I remember the moment at the first developer conference when Commodore announced that the Amiga was going to be a desktop publishing machine. At the time, there was no desktop publishing software available. People asked, "Why desktop publishing?" It seemed an empty pronouncement, made with little awareness of the developing Amiga software market. A year later, the market is not swamped with desktop publishing tools, but video tools have grown in prominence, as many developers predicted.

Commodore will not lose face by admitting that the Macintosh has dominated the desktop publishing

continued...

scene, and that the Amiga is not quite ready to compete with it head-to-head. Without more Postscript support, and much more software, the Amiga will always be a "wanna-be" when it comes to desktop publishing. It would make more sense if Commodore had large booths at video enthusiast shows as well as shows like Seybold.

Palo Alto show

After the Seybold conference, I traveled to Palo Alto for a small show hosted by the local Amiga dealer, Computer Attic. For a small show, it was well-populated with Amiga developers. Mimetics showed their frame buffer and genlock. Neither is shipping now, but they said they hope to have them out in the next few weeks.

Microbotics showed the multifunction daughter board for their Starboard memory expansion. The daughter board fits inside the Starboard case. It has a battery-backed clock and a socket for a 68881 numeric coprocessor. I hope to review this in the next issue. The coprocessor can significantly increase the speed of floating point number calculations. The Amiga operating system has some support for this device, but few existing programs are able to see the speedup offered by this board. This is expected to change as the 68881 coprocessor becomes more popular as an Amiga peripheral.

Microbotics also showed an adapter that lets you mount a Starboard in the Amiga 2000. By removing the Starboard from its plastic case, and mating it with an adapter board, it will fit in the Amiga 2000 box. By fortunate design, Microbotics has peripherals for the entire Amiga line. Their MAS-20 hard disk connects on the parallel port, so it works on the Amiga 500 and 2000 as well as the 1000. They are considering an adapter that fits the Starboard to the Amiga 500.

Projected sales

The latest issue of Amiga Mail had some interesting statistics. Amiga Mail is the newsletter sent to every registered Amiga developer. According to Amiga Mail, about 150,000 Amiga 1000s have been sold worldwide. This number agrees with the best information I've been able to gather from other sources. As of mid-September, as I write this, my best estimates show about eight to ten thousand Amiga 500s sold.

Amiga Mail predicts 300,000 Amiga 500s sold worldwide by the end of the year, and about 70,000 Amiga 2000s in the same time. I think this is a little high. Another source that I consider reliable agrees on the projections for the Amiga 2000, but claims only 150,000 Amiga 500s will be sold by year's end. The emphasis in Amiga Mail is obvious. Commodore wants developers to get excited about the Amiga 500. If only the low sales estimates are true, then the installed base of Amiga machines will double worldwide in the next six months.

The Amiga Mail emphasized the "worldwide" part, too. This issue was chock full of information about making programs compatible with the PAL video standard used outside the United States. A large proportion of those worldwide sales will be to Europe, particularly West Germany.

If you would like to see Amiga Mail, it is available to non-developers for $20 a year. Send your request to Lauren Brown at Commodore, 1200 Wilson Drive, West Chester Pennsylvania 19380. If you register as a developer, the newsletter is included in your fee.

Amiga 2000 ships

The Amiga 2000 has finally shipped. The first 800 units had some trouble in the video circuitry that made the display look fuzzy. Commodore quickly diagnosed the trouble and sent a memo to all authorized service centers. More than 800 machines have been sent out, so the new machines do not have this problem. Only the first 800 were affected, those with serial numbers less than 1006679.

Changes

You may have noticed a change in the masthead of Amazing Computing. I am no longer serving as technical editor. I decided to step down because of my increased financial involvement with Amiga companies. To set the record straight, in recent weeks I served as a paid consultant for Gold Disk and New Tek, and may do work for other companies in the future.

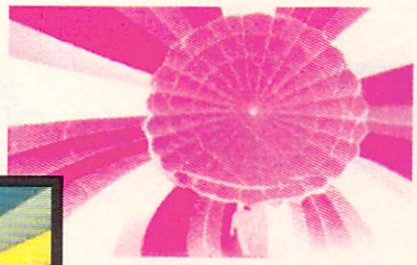
To prevent any charges of conflict of interest, I will avoid writing reviews of their products, or the products of their competitors. Any future comments or criticism (in the pure sense) about these products will include this disclaimer. I believe I can remain objective in my comments about the Amiga marketplace. After all, these companies want to hire an independent mind, not a parrot of their own perceptions.

On top of all this, I have started my own software company, called Syndesis. I won't pull a Pournelle and plug my own product in my column. You'll have to find the Syndesis advertisements on your own.

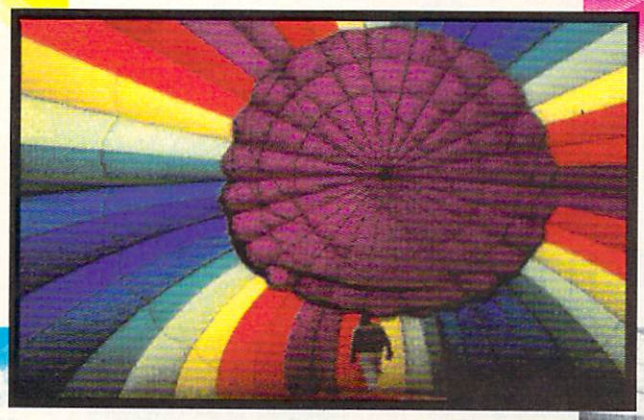
I will continue to write this column and bring you reports from Amiga shows. Next month's coverage will include three shows: the Home Computing Show in San Bruno California, the Commodore Show in Anaheim California, and the first Amiga-specific show, the AmiExpo in New York. Needless to say, I won't be getting much sleep.

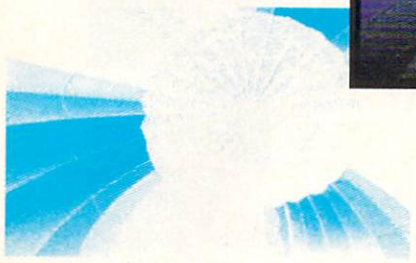
•AC•

100% Yellow

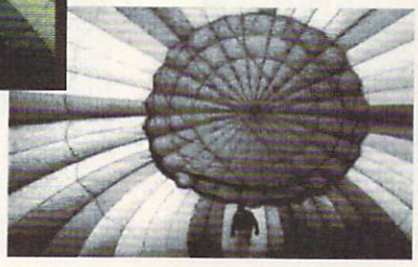


100% Magenta



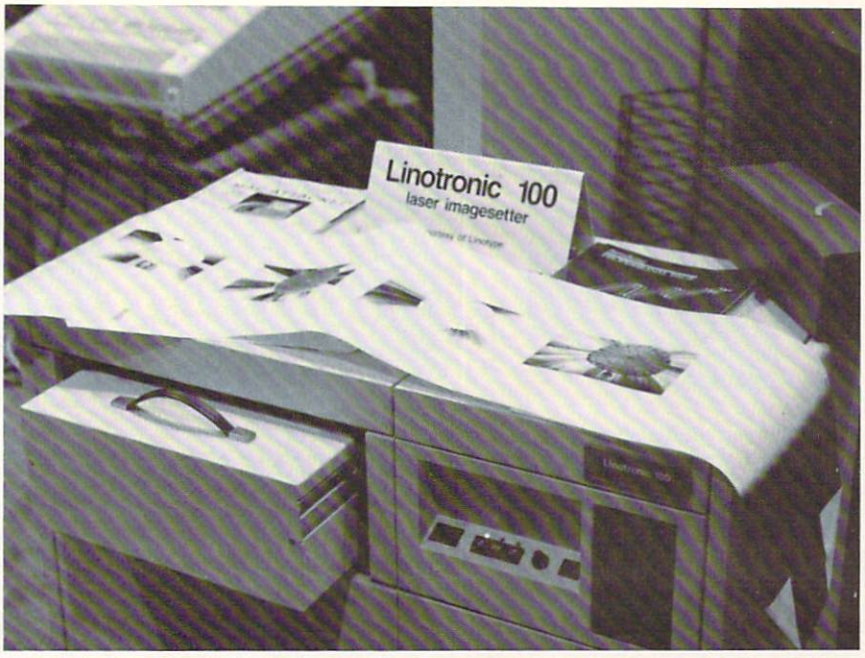


100% Cyan



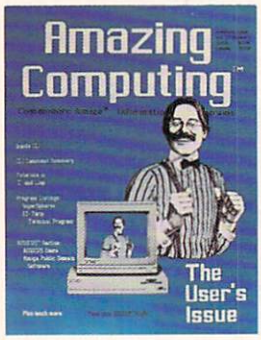
100% Black

These images are examples of a four-color separation from Professional Page. It is a 320 by 400 pixel HAM image digitized with Digi-View. It was printed on a 1200 DPI Linotronic Imagesetter.



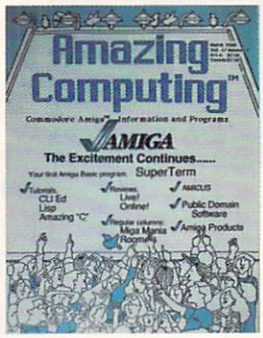
The Linotronic L100 Laser Imagesetter

Expanding Reference



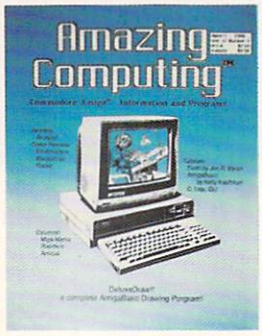
Volume 1 Number 1 Premiere 1986

Super Spheres By Kelly Kauffman An ABasic Graphics prog.
Data Virus By J Foust A disease may attack your Amiga!
EZ-Term by Kelly Kauffman An ABasic Terminal program
Miga Mania by P. Kivolowitz Programming fixes & mouse care
Inside CLI by G. Musser a guided insight into the AmigaDos™ CLI
CLI Summary by G. Musser Jr. A list of CLI commands
AmigaForum by B. Lubkin Visit Compuserve's Amiga SIG
Commodore Amiga Development Program by D. Hicks
Amiga Products A listing of present and expected products



Volume 1 Number 2 March 1986

Electronic Arts Comes Through A review of software from EA
Inside CLI: part two G. Musser Investigates CLI & ED
A Summary of ED Commands
Live! by Rich Miner A review of the Beta version of Live!
Online and the CTS Fabite 2424 ADH Modem by J. Foust
Superterm V 1.0 By K. Kauffman A term. prog. in Amiga Basic
A Workbench "More" Program by Rick Wirch
Amiga BBS numbers



Volume 1 Number 3 April 1986

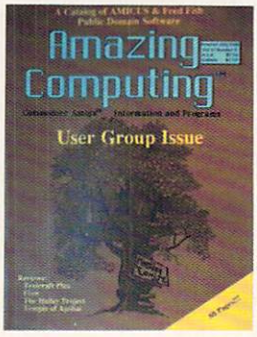
Analyze! a review by Ernest Viveros
Reviews of Racter, Barstaccas and Mindshadow
Forth! The first of our on-going tutorial
Deluxe Draw!! by R. Wrch An Amiga Basic art program
Amiga Basic, A beginners tutorial
Inside CLI: part 3 by George Musser George gives us PIPE

Volume 1 Number 4 May 1986

SkyFox and Artic fox Reviewed
Build your own 5 1/4 Drive Connector By Ernest Viveiros
Amiga Basic Tips by Rich Wirch
Scrimper Part One by P. Kivolowitz prog to print Amiga screen
Microsoft CD ROM Conference by Jim O'Keane
Amiga BBS Numbers

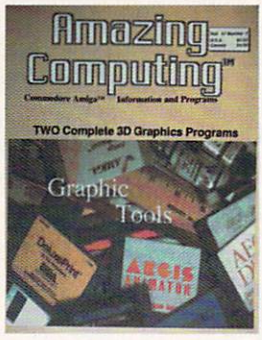
Volume 1 Number 5 1986

The HSI to RGB Conversion Tool
by S. Pietrowicz Color manipulation in BASIC
AmigaNotes by Rick Rae The first of the Amiga music columns
Sidecar A First Look by John Foust A first "under the hood"
John Foust Talks with R. J. Mical at COMDEX™
How does Sidecar affect the Transformer
an interview with Douglas Wyman of Simile
The Commodore Layoffs by J. Foust A look Commodore "cuts"
Scrimper Part Two by Perry Kivolowitz
Marauder reviewed by Rick Wirch
Building Tools by Daniel Kary



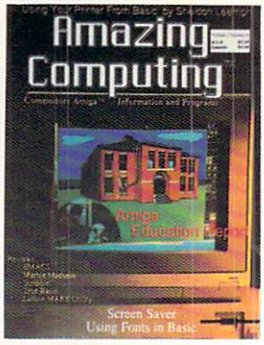
Volume 1 Number 6 1986

Temple of Apshai Trilogy reviewed by Stephen Pietrowicz
The Halley Project: A Mission to our Solar System
reviewed by Stephen Pietrowicz
Flow: reviewed by Erv Bobo
Textcraft Plus a First Look by Joe Lowery
How to start your own Amiga User Group by William Simpson
Amiga User Groups
Mailing List by Kelly Kauffman a basic mail list program
Pointer Image Editor by Stephen Pietrowicz
Scrimper: part three by Perry Kivolowitz
Fun With the Amiga Disk Controller by Thom Sterling
Optimize Your AmigaBasic Programs for Speed by Pietrowicz



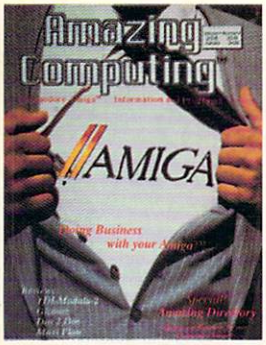
Volume 1 Number 7 1986

Aegis Draw: CAD comes to the Amiga by Kelly Adams
Try 3D by Jim Meadows an introduction to 3D graphics
Aegis Images/ Animator: a review by Erv Bobo
Deluxe Video Construction Set reviewed by Joe Lowery
Window requesters in Amiga Basic by Steve Michel
ROT by Colin French a 3D graphics editor
"I C What I Think" Ron Peterson with a few C graphic progs
Your Menu Sir! by B Catley program Amiga Basic menues
IFF Brush to AmigaBasic 'BOB' Basic editor by M Swinger
Linking C Programs with Assembler Routines on the Amiga
by Gerald Hull



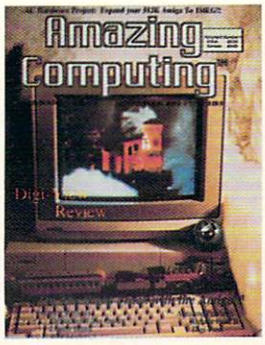
Volume 1 Number 8 1986

The University Amiga By G.Gamble
Amiga's inroads at Washington State University
MicroEd a look at a one man army for the Amiga
MicroEd, The Lewis and Clark Expedition reviewed Frizelle
Scribble Version 2.0 a review
Computers in the Classroom by Robert Frizelle
Two for Study by Frizelle Discovery & TheTalking Coloring Book
True Basic reviewed by Brad Grier
Using your printer with the Amiga
Marble Madness reviewed by Stephen Pietrowicz
Using Fonts from AmigaBasic by Tim Jones
Screen SaVer by P. Kivolowitz A monitor protection prog. in C
Lattice MAKE Utility reviewed by Scott P. Evernden
A Tale of Three EMACS by Steve Poling
.bmap File Reader in Amiga Basic by T Jones



Volume 1 Number 9 1986

Instant Music Reviewed by Steve Pietrowicz
Mindwalker Reviewed by Richard Knepper
The Alegra Memory Board Reviewed by Rich Wirch
TxEd Reviewed by Jan and Cliff Kent
Amazing Directory A guide to the sources and resources
Amiga Developers A listing of Suppliers and Developers
Public Domain Catalog A listing of Amicus and Fred Fish PDS
Dos 2 Dos review R. Knepper
Transfer files from PC/MS-DOS and AmigaBasic
MaxiPlan review by Richard Knepper The Amiga Spreadsheet
Gizmoz by reviewed by Peter Wayner Amiga extras!
The Loan Information Program by Brian Catley
basic prog. to for your financial options
Starting Your Own Amiga Related Business by W. Simpson
Keep Track of Your Business Usage for Taxes by J. Kummer
The Absoft Amiga Fortran Compiler reviewed by R A. Reale
Using Fonts from AmigaBasic, Part Two by Tim Jones
68000 Macros on the Amiga by G. Hull Advance your ability.
TDI Modla-2 Amiga Compiler review by S Faiwize

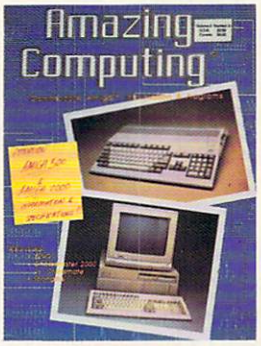


Volume 2 Number 1 1987

What Digi-View Is... Or, What Genlock Should Be! by J. Foust
AmigaBasic Default Colors by Bryan Catley
AmigaBasic Titles by Bryan Catley
A Public Domain Modula-2 System reviewed by Warren Block
One Drive Compile by Douglas Lovell
Using Lattice C with a single drive system
A Megabyte Without Megabucks by Chris Irving
An Internal Megabyte upgrade
Digi-View reviewed by Ed Jakober
Defender of the Crown reviewed by Keith Conforti
Leader Board reviewed by Chuck Raudonis
Roundhill Computer System's PANEL reviewed by Ray Lance
Digi-Paint....by New Tek previewed by John Foust
Deluxe Paint II ....from Electronic Arts previewed by J. Foust

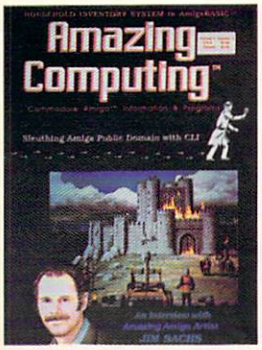
Volume 2 Number 2 1987

The Modem by Josph L. Rothman efforts of a BBS Sysop
MacroModem reviewed by Stephen R. Pietrowicz
GEMINI or "It takes two to Tango" by Jim Meadows
Gaming between machines
BBS-PC! reviewed by Stephen R. Pietrowicz
The Trouble with Xmodem by Joseph L. Rothman
The ACO Project....Graphic Teleconferencing on the Amiga by S. R. Pietrowicz
Flight Simulator II....A Cros Country Tutorial by John Rafferty
A Disk Librarian in AmigaBASIC by John Kennan
Creating and Using Amiga Workbench Icons by C. Hansel
AmigaDOS version 1.2 by Clifford Kent
The Amazing MIDI Interface build your own by Richard Rae
AmigaDOS Operating System Calls and
Disk File Management by D. Heynie
Working with the Workbench by Louis A. Mamakos Prog in C



Volume 2 Number 3

The Amiga 2000™ by J Foust
A First look at the new, high end Amiga™
The Amiga 500™ by John Foust
A look at the new, low priced Amiga
An Analysis of the New Amiga PCs by J. Foust
Speculation on the New Amigas
Gemini Part II by Jim Meadows
The concluding article on two-player games
Subscripts and Superscripts in AmigaBASIC by Ivan C. Smith
The Winter Consumer Electronics Show by John Foust
AmigaTrix by W. Block Amiga™ shortcuts
Intuition Gadgets by Harriet Maybeck Tolly
A journey through gadget-land, using C
Shanghai reviewed by Keith M. Conforti
Chessmaster 2000 & Chessmate reviewedby Edwin V. Apel, Jr.
Zing! from Meridian Software reviewed by Ed Bercovitz
Forth! by Jon Bryan Get stereo sound into your Forth programs.
Assembly Language on the Amiga™ by Chris Martin
Roomers by theBandito Genlocks are finally shipping, & MORE!!!
AmigaNotes by R. Rae Hum Busters... "No stereo? Y not?...
The AMICUS Network by J. Foust
CES, user group issues and Amiga Expo*

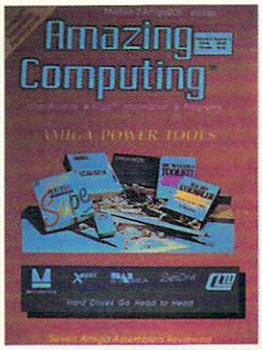


Volume 2 Number 4 1987

Amazing Interviews Jim Sachs by S. Hull Amiga Artist
The Mouse That Got Restored by Jerry Hull and Bob Rhode
Sleuthing Public Domain Disks with CLI by John Foust
Highlights from the San Francisco Commodore Show by S Hul
Speaker Sessions: San Francisco Commodore Show H Tolly
The Household Inventory System in AmigaBASIC™ by B Catley
Secrets of Screen Dumps by Natkun Okun
Using Function Keys with MicroEmacs by Greg Douglas
Amigatrix II by Warren Block More Amiga shortcuts
Basic Gadgets by Brian Catley Create gadget functions
Gridiron reviewed by K. Conforti Real football for the Amiga
Star Fleet I Version 2.1 reviewed by J. Tracy Amigain Space
The TIC reviewed by J. Foust Battery powered Clock Calendar
Metascope review by H. Tolly An easy-to-use debugger

Volume 2 Number 5 1987

The Perfect Sound Digitizer review by R. Battle
The Future Sound Digitizer by W. Block Applied Vision's SD
Forth! by J. Bryancomparing JForth and Multi-Forth.
Basic Input by B. Catley AmigaBASIC input routine for use in all your programs.
Writing a SoundScape Module in C by T. Fay Programming with MIDI, Amiga and SoundScape by SoundScape author.
Programming in 68000 Assembly Language by C. Martin Continuing with Counters & Addressing Modes.
Using FutureSound with AmigaBASIC by J. Meadows
AmigaBASIC Programming utility with real, digitized STEREO
AmigaNotes by R. Rae A review of Mimetics SoundScape Sound Sampler.
More AmigaNotes by R. Rae
A further review of Sunrize's Perfect Sound .
Waveform Workshop in AmigaBASIC by J. Shields edit & save waveform for use in other AmigaBASIC programs.
The Mimetics Pro MIDI Studio by Sullivan, Jeffery
A review of Mimetics' music editor/player.
Intuition Gadgets Part II by H. MaybeckTolly Boolean gadgets provide the user with an on/off user interface.



Volume 2 Number 6 1987

Forth! by J. Bryan Access resources in the ROM Kernal.
The Amazing Computing Hard Disk Review by J. Foust & S. Leemon In-depth looks at the C Ltd. Hard Drive, Microbotics' MAS-Drive20, Byte by Byte's PAL Jr., Supra's 4x4 Hard Drive and Xebec's 9720H Hard Drive. Also, a look at disk driver software currently under development.

Modula-2 AmigaDOS™ Utilities by S. Faiwiszewsk A Calls to AmigaDOS and the ROM kernal.
Amiga Expansion Peripheral by J. Foust
Explanation of Amiga expansion peripherals.
Amiga Technical Support by J. Foust
How and where to get Amiga tech support.
Goodbye Los Gatos by J. Foust Closing Los Gatos.
The Amicus Network by J. Foust West Coast Computer Faire.
Metacomco Shell and Toolkit by J. Foust A review
The Magic Sac by J. Foust Run Mac programs on your Amiga.
What You Should Know Before Choosing an Amiga 1000 Expansion Device by S. Grant
7 Assemblers for the Amiga by G. Hull Choose your assembler
High Level Shakeup Replaces Top Management at Commodore by S. Hull
Peter J. Baczor" by S. Hull Manager at CBM gives an inside look
Logistix A review by Richard Knepper
Organize! by A review Richard Knepper database.
68000 Assembly Language Programming on the Amiga by Chris Martin
Superbase Personal Relational Database by Ray McCabe
AmigaNotes by Rae, Richard A look at FutureSound
Commodore Shows the Amiga 2000 and 500 at the Boston Computer Society by H Maybeck Tolly

Volume 2, Number 7 1987

New Breed of Video Products by John Foust...
Very Vivid! by Tim Grantham...
Video and Your Amiga by Oran Sands III
Amigas & Weather Forecasting by Brenden Larson
A-Squared and the Live! Video Digitizer by John Foust.
Aegis Animator Scripts and Cel Animation by John Foust
Quality Video from a Quality Computer by Oran Sands III.
Is IFF Really a Standard? by John Foust..
Amazing Stories and the Amiga™ by John Foust.
All about Printer Drivers by Richard Bielak
Intuition Gadgets by Harriet Mayback Tolley.
Deluxe Video 1.2 by Bob Eller
Pro Video CG1 by Oran Sands III.
Digi-View 2.0 Digitizer/Software by Jennifer M. Janik
Prism HAM Editor from Impulse by Jennifer M. Janik
Easy! drawing tablet by John Foust..
CSA's Turbo-Amiga Tower by Alfred Aburto
68000 Assembly Language by Chris Martin.

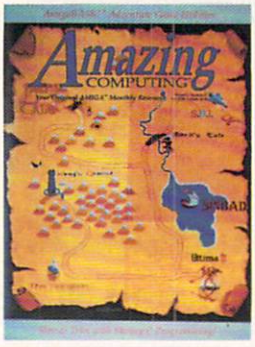
Your Resource to the Commodore Amiga™

The above phrase is much more than empty words. The pages of Amazing Computing™ are filled with articles on technical operations and procedures, basic use, and just-plain-fun. The growing library of Amazing Back Issues contains articles ranging from building your own IBM Disk controller, to setting up your own startup sequence. Amazing Computing™ has repeatedly been the first magazine to offer users solid, in-depth reviews and hands-on articles for their Amigas.

Amazing Computing™ was the first magazine to document CLI.
Amazing Computing™ was the first to show Sidecar™ from COMDEX™ in full detail.
Amazing Computing™ was the first to document a 5 1/4 drive connector.
Amazing Computing™ was the first with a 1 Meg Amiga upgrade hardware project!
Amazing Computing™ was the first magazine to offer serious programming examples and help.
Amazing Computing™ was the first magazine to offer Public Domain Software at reasonable prices.
Amazing Computing™ was the first magazine with the user in mind!

From the Beginning

Since February 1986, Amazing Computing™ has been providing users with complete information for their Amigas. This vast collection of programs and information is still available through our back issues. From the Premiere issue to the present, AC has been packed with Amiga insights that any user will find informative.

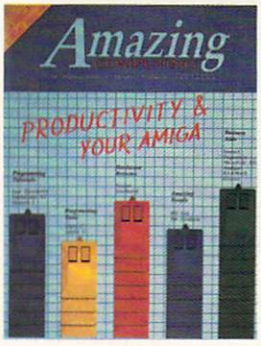


Volume 2, Number 8 1987

This month Amazing Computing™ focuses on entertainment packages for the Amiga. Amazing game reviews...

SDI, Earl Weaver Baseball, Portal, The Surgeon, Little Computer People, Sinbad, StarGlider, King's Quest I,II and III, Faery Tale Adventure, Ultima III, Facets of Adventure, Video Vegas and Bard's Tale.

Plus Amazing monthly columns... Amiga Notes, Roomers, Modula-2, 68000 Assembly Language and The Amicus Network.
Disk-2-Disk by Matthew Leeds
The ColorFonts Standard by John Foust
Skinny C Programs by Robert Riemersma, Jr.
Hidden Messages In Your Amiga™ by John Foust
The Consumer Electronics Show and Comdex by J Foust



Volume 2 Number 9 1987

Analyze 2.0 reviewed by Kim Schaffer
Impact Business Graphics review by Chuck Raudonis
Microfiche Filer review by Harv Laser
Pagesetter review by Rick Wrch
Gizmoz Productivity Set 2.0 review by Bob Eller
Kickwork review by Harv Laser
Diga Telecommunications Package review by Steve Hull
Mouse Time and Timesaver review by John Foust
Insider Memory Expansion review by James O'Keane
Microbotics Starboard-2 review by S. Fawiszewski
Leather Goddesses of Phobos
reviewed by Harriet Maybeck-Tolly
Lattice C Compiler Version 3.10 reviewed by Gary Sarff
Manx 3.4a Update reviewed by John Foust
AC-BASIC reviewed by Sheldon Leemon
AC-BASIC Compiler an alternative comparison by B Catley
Modula-2 Programming S Fawiszewski
Raw Console Device Events
Directory Listings Under AmigaDOS by Dave Haynie
AmigaBASIC Patterns by Brian Catley
Programming with Soundscape
Todor Fay manipulates samples
Bill Volk, Vice-President Aegis Development,
interviewed by Steve Hull
Jim Goodnow, Developer of Manx 'C'
interview by Harriet M Tolly

To Be Continued.....

$4.00 each!

Our Back Issue price is still $4.00 per issue!

(Foreign orders, please add $1.00 per issue for Postage & Handling. All payments must be made by check or money order in U.S. funds drawn on a U.S. Bank.)

Limited Supply

Unfortunately, nothing lasts forever. The availabilty of some of our Back Issues is definitely limited. Please complete your Amazing Computing™ library today, while these issues are still available. Please complete the order form in the rear of this issue and mail with check or money order to:

Back Issues
PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722

(Please allow 4 to 6 weeks for delivery)

The AMICUS & Fred Fish

Public Domain Software Library

This software is collected from user groups and electronic bulletin boards around the nation. Each Amicus disk is nearly full, and is fully accessible from the Workbench. If source code is provided for any program, then the executable version is also present. This means that you don't need the C compiler to run these programs. An exception is granted for those programs only of use to people who own a C compiler.

The Fred Fish disk are collected by Mr. Fred Fish, a good and active friend of the Amiga.

Note: Each description line below may include something like 'S-O-E-D', which stands for 'source, object file, executable and documentation'. Any combination of these letters indicates what forms of the program are present. Basic programs are presented entirely in source code format.

AMICUS Disk 1

ABasic programs: Graphics

3DSolids	3d solids modeling prog. w/sample data files
Blocks	draws blocks
Cubes	draws cubes
Durer	draws pictures in the style of Durer
FScape	draws fractal landscapes
Hidden	3D drawing program, w/ hidden line removal
JPad	simple paint program
Optical	draw several optical illusions
PaintBox	simple paint program
Shuttle	draws the Shuttle in 3d wireframe
SpaceArt	graphics demo
Speaker	speech utility
Sphere	draws spheres
Spiral	draws color spirals
ThreeDee	3d function plots
Topography	artifical topography
Wheels	draws circle graphics
Xenos	draws fractal planet landscapes

ABasic programs: Tools

AddressBook	simple database program for addresses
CardFile	simple card file database program
Demo	multiwindow demo
KeyCodes	shows keycodes for a key you press
Menu	run many ABasic programs from a menu
MoreColors	way to get more colors on the screen at once, using aliasing
shapes	simple color shape designer SpeakIt speech and narrator demo

ABasic programs: Games

BrickOut	classic computer brick wall game
Othello	also known as 'go'
Saucer	simple shoot-em-up game
Speling	simple talking spelling game
ToyBox	selectable graphics demo

ABasic programs: Sounds

Entertainer	plays that tune
HAL9000	pretends it's a real computer
Police	simple police siren sound
SugarPlum	plays "The Dance of the Sugarplum Fairies"

C programs:

ATerm	simple terminal program, S-E
cc	aid to compiling with Lattice C
decvnt	opposite of CONVERT for cross developers
Dotty	source code to the 'dotty' window demo
echox	unix-style filename expansion, partial S,O-D
fasterfp	explains use of fast-floating point math
FixDate	fixes future cates on all files on a disk, S-E
freedraw	simple Workbench drawing prog.,S-E
GfxMem	graphic memory usage indicator, S-E
Grep	searches for a given string in a file with
docs.	ham shows off the hold-and-modify method of color generation
IBM2Amiga	fast parallel cable transfers between an IBM and an Amiga
Mandel	Mandelbrot set program, S-E
moire	patterned graphic demo, S-E
objfix	makes Lattice C object file symbols visible to Wack, S-E
quick	quick sort strings routine
raw	example sample window I/O
setface	turns on interlace mode, S-E
sparks	qix-type graphic demo, S-E

Other executable programs:

SpeechToy	speech demonstration
WhichFont	displays all available fonts

Texts:

68020	describes 68020 speedup board from CSA
Aliases	explains uses of the ASSIGN command
Bugs	known bug list in Lattice C 3.02
CLICard	reference card for AmigaDOS CLI
CLICommands	guide to using the CLI
Commands	shorter guide to AmigaDOS CLI commands
EdCommands	guide to the ED editor
Filenames	AmigaDOS filename wildcard conventions
HalfBright	explains rare graphics chips that can do more colors
ModemPins	description of the serial port pinout
RAMdisks	tips on setting up your RAM: disk
ROMWack	tips on using ROMWack
Sounds	explanation of Instrument demo sound file format
Speed	refutation of Amiga's CPU and custom chip speed
WackCmds	tips on using Wack

AMICUS Disk 2

C programs:

alib	AmigaDOS object library manager , S-E
ar	text file archive program, S-E
fixobj	auto-chops executable files
shell	simple CLI shell, S-E
sq, usq	file compression programs, S-E
YachtC	a familiar game, S-E
Make	a simple 'make' programming utility, S-E
Emacs	an early version of the Amiga text editor, S-E-D

Assembler programs:

bsearch.asm	binary search code
qsort.asm	Unix compatible qsort() function, source and C test program
setjmp.asm	setjmp() code for Lattice 3.02
SVprintf	Unix system V compatible printf()
trees.o	Unix compatible tree() function, O-D

(This disk formerly had IFF specification files and examples. Since this spec is constantly updated, the IFF spec files have been moved to their own disk in the AMICUS collection.)

John Draper Amiga Tutorials:

Animate	describes animation algorithms
Gadgets	tutorial on gadgets
Menus	learn about Intuition menus

AMICUS Disk 3

C programs:

Xref	a C cross-reference gen., S-E
6bitcolor	extra-half-bright chip gfx demo, S-E
Chop	truncate (chop) files down to size, S-E
Cleanup	removes strange characters from text files
CR2LF	converts carriage returns to line feeds in Amiga files, S-E
Error	adds compile errors to a C file, S
Hello	window ex. from the RKM, S
Kermit	generic Kermit implementation, flakey, no terminal mode, S-E
Scales	sound demo plays scales, S-E
SkewB	Rubik cube demo in hi-res colors, S-E

AmigaBasicProgs(dir)

Automata	cellular automata simulation
CrazyEights	card game
Graph	function graphing programs
WitchingHour	a game

ABasiC programs:

Casino	games of poker, blackjack, dice, and craps
Gomoku	also known as 'othello'
Sabotage	sort of an adventure game

Executable programs:

Disassem	a 68000 disassembler, E-D
DpSide	shows a given set of IFF pictures, E-D
Arrange	a text formatting program, E-D

Assembler programs:

Argoterm	terminal program with speech and Xmodem, S-E

AMICUS Disk 4 Files from the original Amiga Technical BBS

Note that some of these files are old, and refer to older versions of the operating system. These files came from the Sun system that served as Amiga technical support HQ for most of 1985. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

Complete and nearly up-to-date C source to 'image.ed', an early version of the Icon Editor. This is a little flaky, but compiles and runs.

An Intuition demo, in full C source, including files: demomenu.c, demomenu2.c, demoreq.c, getascii.c, idemo.c, idemo.guide, idemo.make, idemoall.h, nodos.c, and bowtie.c

addmem.c	add external memory to the system
bobtest.c	example of BOB use
consoleIO.c	console IO example
creaport.c	create and delete ports
creastdi.c	create standard I/O requests
creatask.c	creating task examples
diskio.c	example of track read and write
dotty.c	source to the 'dotty window' demo
dualplay.c	dual playfield example
flood.c	flood fill example
freemap.c	old version of 'freemap'
getpsols.c	tools for VSprites and BOBs
gfxmem.c	graphic memory usage indicator
hello.c	window example from RKM
inputdev.c	adding an input handler to the input stream
joystik.c	reading the joystick
keybd.c	direct keyboard reading
layertes.c	layers examples
mousport.c	test mouse port
ownlib.c, ownlib.asm	example of making your own library with Lattice
paratest.c	tests parallel port commands
sertest.c	tests serial port commands
sersamp.c	example of serial port use
printtr.c	sample printer interface code
prtbase.h	printer device definitions
regntes.c	region test program
setface.c	source to interlace on/off program
setparallel.c	set the attributes of the parallel port
SetSerial.c	set the attributes (parity, data bits, baud rate, etc.) for the serial port
singplay.c	single playfield example
speechtoy.c	source to narrator and phonetics demo
timedely.c	simple timer demo
timer.c	exec support timer functions
timrstuf.c	more exec support timer functions
WhichFont.c	loads and displays all available system fonts

process.i and prtbase.i assmebler include files:

autorqstr.txt	warnings of deadlocks with autorequesters
consoleIO.txt	copy of the RKM console I/O chapter
diskfont.txt	warning of disk font loading bug
fullfunc.txt	list of #defines, macros, functions
inputdev.txt	preliminary copy of the input device chapter

License information on Workbench distribution license

printer pre-release copy of the chapter on printer drivers, from RKM 1.1 v11fd.txt 'diff of .fd file changes from version 1.0 to 1.1 v28v1.diff 'diff' of include file changes from version 26 to 1.0

AMICUS Disk 5 Files from the Amiga Link / Amiga Information Network

Note that some of these files are old, and refer to older versions of the operating system. These files are from Amiga Link. For a time, Commodore supported Amiga Link, aka AIN, for online developer technical support. It was only up and running for several weeks. These files do not carry a warranty, and are for educational purposes only. Of course, that's not to say they don't work.

A demo of Intuition menus called 'menudemo', in C source

whereis.c	find a file searching all subdirectories
bobtest.c	BOB programming example
sweep.c	sound synthesis example

Assembler files:

mydev.asm	sample device driver
mylib.asm	sample library example
mylib.i	
mydev.i	
asmsupp.i	
macros.i	assembler include files

Texts:

amigatricks	tips on CLI commands
extdisk	external disk specification
gameport	game port spec
parallel	parallel port spec
serial	serial port spec
v1.1update	list of new features in version 1.1
v1.1fn.txt	'diff' of include file changes from version

Files for building your own printer drivers, including dospecial.c, epsondata.c, init.asm, printer.c, printer.link, printertag.asm, render.c, and wait.asm. This disk does contain a number of files describing the IFF specification. These are not the latest and greatest files, but remain here for historical purposes. They include text files and C source examples. The latest IFF spec is elsewhere in this library.

AMICUS Disk 6 IFF Pictures

This disk includes the DPSlide program, which can view a given series of IFF pictures, and the 'showpic' program, which can view each file at the click of an icon. The pictures include a screen from ArtcFox, a Degas dancer, the guys at Electronic Arts, a gorilla, horses, King Tut, a lighthouse, a screen from Marble Madness, the Bugs Bunny Martian, a still from an old movie, the Dire Straits moving company, a screen from Pinball Contruction Set, a TV newcaster, the PaintCan, a world map, a Porsche, a shuttle mission patch, a tyrannosaurus rex, a planet view, a VISA card, and a ten-speed.

AMICUS Disk 7 DigiView HAM demo picture disk

This disk has pictures from the DigiView hold-and-modify video digitizer. It includes the ladies with pencils and lollypops, the young girl, the bulldozer, the horse and buggy, the Byte cover, the dictionary page, the robot and Robert. This includes a program to view each picture separately, and all together as separate, slidable screens. The 'seeibm' program, to turn any screen into an IFF picture.

AMICUS Disk 8

C programs:

Browse	view text files on a disk, using menus S-E-D
Crunch	removes comments and white space from C files, S-E
IconExec	EXECUTE a series of commands from Workbench S-E
PDScreen Dump	dumps Rastport of highest screen to printer
SetAltemate	sets a second image for an icon, when clicked once S-E
SetWindow	makes windows for a CLI program to run under Workbench S-E
SmallClock	a small digital clockin a window menu bar
Scrimper	the screen printer in the fourth AC S-E

Amiga Basic Programs:

(Note: Many of these programs are present on AMICUS Disk 1. Several of these were converted to Amiga Basic, and are included here.)

AddressBook	a simple address book database
Ball	draws a ball
Cload	program to convert Compuserve hex files to binary, S-D
Clue	the game, Intuition driven
ColorArt	art drawing program
DeluxeDraw	the drawing program in the 3rd AC, S-D
Eliza	conversational computer psychologist
Othello	the game, as known as 'go'
RatMaze	3D ratmaze game
ROR	boggling graphics demo
Shuttle	draws 3D pictures of the space shuttle
Speling	simple spelling program
YoYo	wierd zero-gravity yo-yo demo, tracks yo-yo to the mouse

Executable programs:

3Dcube	Modula-2 demo of a rotating cube
AltIcon	sets a second icon image, displayed when the icon is clicked
AmigaSpell	a slow but simple spell checker, E-D
arc	the ARC file compression progra must-have for telecom, E-D
Bertrand	graphics demo
diskSalvage	prog. to rescue trashed disks, E-D
KwikCopy	a quick but nasty disk copy program: ignores errors, E-D
LibDir	lists hunks in an object file E-D
SaveILBM	saves any screen as IFF pic.E-D ??
ScreenDump	shareware screen dump prog, E only
StarTerm	version 2.0, term program, XmodemE-D

Texts:

LatticeMain	tips on fixing _main.c in Lattice
GDiskDrive	make your own 5 1/4 drive
GuruMed	explains the Guru numbers
Lat3.03bugs	bug list of Lattice C version 3.03
MForgeRev	user's view of the MicroForge HD
PrintSpooler	EXECUTE-based print spool prog.

.BMAP files:

These are the necessary links between Amiga Basic and the system libraries. To take advantage of the Amiga's capabilities in Basic, you need these files. BMAPs are included for 'dist', 'console', 'diskfont', 'exec', 'icon', 'intuition', 'layers', 'mathffp', 'mathieeedoubas', 'mathieeesingbas', 'mathtrans', 'potgo', 'timer' and 'translator'.

AMICUS Disk 9

Amiga Basic Programs:

FlightSim	simple flight simulator program
HuePalette	explains Hue, Saturation, & Intensity
Requester	ex. of requesters from Amiga Basic
ScrollDemo	demonstrates scrolling capabilities
Synthesizer	sound program
WorldMap	draws a map of the world

Executable programs:

Boing!	latest Boing! demo,with selectable speed,E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time & date
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

Texts:

FnctnKeys	read function keys from Amiga Basic
HackerSin	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
Boing!	latest Boing! demo,with selectablespeed, E
Brush2C	converts an IFF brush to C data instructions, initialization code, E
Brush2Icon	converts IFF brush to an icon, E
Dazzle	graphics demo, tracks to mouse, E
DeciGEL	assembler program for stopping 68010 errors, S-E-D
Klock	menu-bar clock and date display, E
life	the game of life, E
TimeSet	Intuition-based way to set the time date,
EMEmacs	another Emacs, more oriented to word processing, S-E-D
MyCLI	a CLI shell, works without the Workbench, S-E-D

For PDS orders, Please use form on page 128

Texts:

FnctnKeys	explains how to read function keys from Amiga Basic
HackerSh	explains how to win the game 'hacker'
Ist68010	guide to installing a 68010 in your Amiga
PrinterTip	sending escape sequences to your printer
StartupTip	tips on setting up your startup-sequence file
XfrmrReview	list of Transformer programs that work

Printer Drivers:
Printer drivers for the Canon PJ-1080A, the C Itoh Prowriter, an improved Epson driver that eliminates streaking, the Epson LQ-800, the Gemini Star-10, the NEC 8025A, the Okidata ML-92, the Panasonic KX-P10xx family, and the Smith-Corona D300, with a document describing the installation process.

AMICUS Disk 10 **Instrument sound demos**
This is an icon-driven demo, circulated to many dealers. It includes the sounds of an acoustic guitar, an alarm, a banjo, a bass guitar, a boink, a calliope, a car horn, claves, water drip, electric guitar, a flute, a harp arpegio, a kickdrum, a marimba, a organ minor chord, people talking, pigs, a pipe organ, a Rhodes piano, a saxophone, a sitar, a snare drum, a steel drum, bells, a vibrophone, a violin, a wailing guitar, a horse whinny, and a whistle.

AMICUS Disk 11

C programs

drutil	Intuition-based, CLI replacement manager S-E
cpri	shows and adjusts priority of CLI processes, S-E
ps	shows info on CLI processes, S-E
vidtex	displays Compuserve RLE pics, S-E

AmigaBasic programs

pointered	pointer and sprite editor program
optimize	optimization ex ample from AC article
calendar	large, animated calendar, diary and date book program
amortize	loan amortizations
brushtoBOB	converts small IFF brushes to AmigaBasic BOB OBJECTS
grids	draw and play waveforms
hilbert	draws Hilbert curves
madlib	mad lib story generator
mailtalk	talking mailing list program
meadows3D	3D graphics program, from A C™ article
mousetrack	mouse tracking example in hires mode
slot	slot machine game
tictactoe	the game
switch	pachinko-like game
weird	makes strange sounds

Executable programs

cp	unix-like copy command, E
cls	screen clear, S-E
diff	unix-like stream editor uses 'diff' output to fix files
pm	chart recorder performances indicator

Assembler programs

cls	screen clear and CLI arguments example

Modula-2

trails	moving-worm graphics demo
caseconvert	converts Modula-2 keywords to uppercase
Forth	Breshehan circle algorithm example
Analyze	12 templates for the spreadsheet Analyze

There are four programs here that read Commodore 64 picture files. They can translate Koala Pad, Doodle, Print Shop and News Room graphics to IFF format. Getting the files from your C-64 to your Amiga is the hard part.

AMICUS Disk 12

Executable programs

blink	'alink' compatible linker, but faster, E-D
clean	spins the disk for disk cleaners, E-D
epsonset	sends Epson settings to PAR from menu E-D
showbig	view hi-res pics in low-res superbitmap, E-D
speaktime	tell the time, E-D
undelete	undeletes a file, E-D
cnvapldhm	converts Apple][low, medium and high res pictures to IFF, E-D
menued	menu editor produces C code for menus, E-D
quick	quick disk-to-disk nibble copier, E-D
quickEA	copies Electronic Arts disks, removes protection, E-D
txed 1.3	demo of text editor from Microsmiths,E-D

C programs

spin3	rotating blocks graphics demo, S-E-D
popdi	start a new CLI at the press of a button, like Sidekick, S-E-D
vsprite	VSprite example code from Commodore, S-E-D
AmigaBBS	Amiga Basic bulletin board prog., S-D

Assembler programs

star10	makes star fields like Star Trek intro,S-E-D

Pictures

Mount Mandelbrot	3D view of Mandelbrot set
Star Destroyer	hi-res Star Wars starship
Robot	robot arm grabbing a cylinder

Texts

vendors	Amiga vendors, names, addresses
cardco	fixes to early Cardco memory boards
cinclude	cross-reference to C include files
mindwalker	clues to playing the game well
slideshow	make your own slideshows from the Kaleidoscope disk

AMICUS Disk 13

Amiga Basic programs
Routines from Carolyn Schepper of CBM Tech Support, to read and display IFF pictures from Amiga Basic. With documentation. Also included is a program to do screen prints in Amiga Basic, and the newest BMAP files, with a corrected ConvertFD program. With example pictures, and the SaveILBM screen capture program.

Routines to load and play FutureSound and IFF sound files from Amiga Basic, by John Foust for Applied Visions. With documentation and C and assembler source for writing your own libraries, and interfacing C to assembler in libraries. With example sound.

Executable programs

gravity	Sci Amer Jan 86 gravitation graphic simulation, S-E-D

Texts

MIDI	make your own MIDI instrument interface, with documentation and a hi-res schematic picture.

AMICUS Disk 14

Several programs from Amazing Computing issues:

Tools

Dan Kary's	C structure index program, S-E-D

Amiga Basic programs:

BMAP Reader	by Tim Jones
IFFBrush2BOB	by Mike Swinger
AutoRequester	example
DOSHelper	Windowed help system for CLI commands, S-E-D
PETrans	translates PET ASCII files to ASCII files, S-E-D
C Squared	Graphics program from Scientific American, Sept 86, S-E-D
crlf	adds or removes carriage returns from files, S-E-D
dpdecode	decrypts Deluxe Paint, remo
ves copy	protection, E-D
queryWB	asks Yes or No from the user returns exit code, S-E
vc	VisiCalc type spreadsheet, no mouse control, E-D
view	views text files with window and
slider	gadget, E-D
Oing, Sproing,	yaBoing, Zoing are sprite-based
Boing! style	demos, S-E-D

CLIClock, sClock, wClock are window border clocks, S-E-D

Texts
An article on long-persistance phospor monitors, tips on making brushes of odd shapes in Deluxe Paint, and recommendations on icon interfaces from Commodore-Amiga.

AMICUS 15

The C programs include:

'pr'	a file printing utility, which can print files in the background, and with line numbers and control character filtering.
'fm'	displays a chart of the blocks allocated on a disk.
'Ask'	questions an 'execute' file, returns an error code to control the execution in that batch file
'Stat'	an enhanced version of AmigaDOS 'status' command.
'Dissolve'	random-dot dissolve demo displays IFF picture slowly, dot by dot, in a random fashion.
'PopCLI2'	invoke new CLI window at the press of a key.

The executable programs include:

'Form'	file formatting program through the printer driver to select print styles
'DiskCat'	catalogs disks, maintains, sorts,merges lists of disk files
'PSound'	SunRize Industries' sampled sound editor & recorder
'Iconmaker'	makes icons for most programs
'Fractals'	draws great fractal seascapes and mountain scapes.
'3D Breakout'	3D glasses, create breakout in a new dimension
'AmigaMonitor'	displays lists of open files,
memory use,	tasks, devices and ports in use.
'Cosmoroids'	version of 'asteroids' for the Amiga.
'Sizzlers'	high resolution graphics demo written in Modula 2.

Texts:

'ansi.txt'	explains escape sequences the CON: device responds to.
'FKey'	includes template for making paper to sit in the tray at the top of the Amiga keyboard.
'Spawn'	programmer's document from Commodore

Amiga, describes ways to use the Amiga's multitasking capabilities in your own programs.

AmigaBasic programs:

'Grids'	draw sound waveforms, and hear them played.
'Light'	a version of the Tron light-cycle video game.
'MigaSol'	a game of solitaire.
'Stats'	program to calculate batting averages
'Money'	"try to grab all the bags of money that you can."

AMICUS 15 also includes two beautiful IFF pictures, of the enemy walkers from the ice planet in Star Wars, and a picture of a cheetah.

AMICUS 16

'juggler'	demo by Eric Graham, a robot juggler bouncing three mirrored balls, with sound effects. Twenty-four frames of HAM animation are flipped quickly to produce this image. You control the speed of the juggling. The author's documentation hints that this program might someday be available as a product.

IFF pictures

	parodies of the covers of Amiga World and Amazing Computing magazines.

C programs:

'Inputhandler'	example of making an input handler.
'FileZap3'	binary file editing program
'ShowPrint'	displays IFF picture, and prints it.
'Gen'	program indexes and retrieves C structures and variables declared in the Amiga include file system.

Executable Programs:

'FixHunk2'	repairs an executable program file for expanded memory
'ms2smus'	converts Music Studio files to IFF standard 'SMUS' format. I have heard this program might have a few bugs, especially in regards to very long songs, but it works in most cases.
'Missile'	Amiga version of the "Missile Command" video game,

This disk also contains several files of scenarios for Amiga Flight Simulator II. By putting one of these seven files on a blank disk, and inserting it in the drive after performing a special command in this game, a number of interesting locations are preset into the Flight Simulator program. For example, one scenario places your plane on Alcatraz, while another puts you in Central Park

AMICUS 17

Telecommunications disk which contains six terminal programs.

"Comm" V1.33	term prog. with Xmodem, WXmodem,
"ATerm" V7.2	term prog. includes Super Kermit
"VT-100"V2.6	Dave Wecker's VT-100 emulator with Xmodem,Kermit, and scripting
"Amiga Kermit"	V4D(060) port of the Unix C-Kermit
"VTek" V2.3.1	Tektronix graphics terminal emulator based on the VT-100 prog. V2.3 and contains latest 'arc' file compression
"AmigaHost"	V0.9 for Compuserve. Includes RLE graphics abilities & CIS-B file transfer protocol.
"FixHunk"	expansion memory necessity
"FixObj"	removes garbage characters from modem received files
"Txt"	filters text files from other systems to be read by the Amiga E.C.
"addmem"	executeable version for use wth mem expansion article in AC v2..1
'arc'	file documentation and a basic tutorial on un 'arc'ing files
"arcre"	for makeing "arc" files E.C.

AMICUS Disk 18

Logo	Amiga version of the popularcomputer language, wth example programs, E-D
TV'Text	Demo version of the TV'Text character generator
PageSetter	Freely distributable versions of the updated PagePrint and PageIFF programs for the PageSetter desktop publishing package.
FullWindow	Resizes any CLI window using only CLI commands, E-D
Life3d	3-D version of Conway's LIFE program, E-D
Defdisk	CLI utility to re-assign a new Workbench disk, S-E-D
Calendar.WKS	Lotus-compatible worksheet that makes calendars
SetKey	Demo of keyboard key re-programmer, with IFF picture to make function key labels, E-D
VPG	Video pattern generator for aligning monitors, E-D
HP-10C	Hewlett-Packard-like calculator, E-D
SetPrefs	Change the Preferences settings on the fly, in C, S-E-D
StarProbe	Program studies stellar evolution. C source included for Amiga and MS-DOS, S-E-D
ROT	C version of Colin French's AmigaBasic ROT program from Amazing Computing. ROT edits and displays polygons to create three dimensional objects. Up to 24 frames of animation can be created and displayed. E-D
Scat	Like Ing, windows on screen run away from the mouse, E-D
DK	Decays" the CLI window into dust, in Module 2, S-E-D
DropShadow2	Adds layered shadows to Workbench windows, E-D

AMICUS 19

This disk carries several programs from Amazing Computing. The IFF pictures on this disk include the Amiga Wake part T-shirt logo, a sixteen-color hi-res image of Andy Griffith, and five Amiga Live! pictures from the Amazing Stories episode that featured the Amiga.

Solve	Linear equation solver in assembly language, S-E-D
Gadgets	Bryan Catley's AmigaBasicGadbrial,
Household	Bryan Catley's AmigaBasic household inventory program, S-D
Waveform	Jim Shields' Waveform WoAmigaBasic, S-D
DiskLib	John Kennan's AmigaBasic disk librarian program, S-D
Subscripts	Ivan Smith's AmigaBasic subscript example, S-D
String, Boolean	C programs and executables for Harriet Maybeck Tolly's Intuition tutorials, S-E-D
Skinny C	Bob Riemersma's example for making small C programs, S-E-D
COMAL.h	Make C look like COMAL hEader file,
EmacsKey	Makes Emacs function key definitions by Greg Douglas, S-D
AMon 1.1	Snoop on system resource use, E-D
BTE	Bard's Tale character editor, E-D
Size	CLI program shows the size of a given set of files, E-D
WinSize	CLI window utility resizes current window, S-E-D

Disk 20

Compactor, Decoder Steve Michel AmigaBasic tools, S-D

BobEd	BOB and sprite editor written in C,S-E-D
SpriteMasterII	Sprite editor and animator by Brad Kiefer, E-D
BitLab	Bitter chip exploration C program by Tomas Rokicki, S-E-D
FPic	Image processing program by Bob Bush loads and saves IFF images, changes them with several techniques, E-D
Bankn	Complete home banking program, balance your checkbook! E-D
cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory
input.dev	sample input handler, traps key or mouse events
joystick	Shows how to set up the gameportdevice as a joystick.
keyboard	demonstrates direct communications with the keyboard.
layers	Shows use of the layers library
mandelbrot	IFF Mandelbrot program
mouse	hooks up mouse to right joystick port
one.window	console window demo
parallel	Demonstrates access to the parallel port.
printer	opening and using the printer, does a screen dump, not working
print.support	Printer support routines, not working.
proctest	sample process creation code, not working
region	demos split drawing regions
samplefont	sample font with info on creating your own
serial	Demos the serial port
singlePlayfield	Creates 320 x 200 playfield
speechtoy	latest version of cute speech demo
speech.demo	simplified version of speechtoy, with IO requests
text.demo	displays available fonts
timer	demos timer.device use
trackdisk	demos trakcdisk driver

AMICUS 21

Target	Makes each mouse click sound like a gunshot, S-E-D
Sand	Simple game of sand that follows the mouse pointer, E-D
PropGadget	Harriet Maybeck Tolly's proportional gadget example, S-E
EHB	Checks to see if you have extra-half-bright graphics, S-E-D
Piano	Simple piano sound program
CelScripts	Makes cel animation scripts for Aegis Animator, in AmigaBasic

This disk has electronic catalogs for AMICUS disks 1 to 20 and Fish disks 1 to 80. They are viewed with the DiskCat program, included here.

AMICUS 22

Cycles	Light cycle game, E-D
Show_PrintII	Views and prints IFF pictures, including larger than screen
PrtDrvGen2.3	Latest version of a printer driver generator
Animations	VideoScape animations of planes and boing ball
Garden	Makes fractal gardenscapes
BasicSorts	Examples of binary search and insertion sort in AmigaBasic

Fred Fish Public Domain Software

Fred Fish Disk 1:

amigademo	Graphical benchmark for comparing amigas.
amigaterm	simple communications program with Xmodem
balls	simulation of the "kinetic thingy" with balls on strings
colorful	Shows off use of hold-and-modify mode.
dhrystone	Dhrystone benchmark program.
dotty	Source to the "dotty window" demo on the Workbench disk.
freedraw	A small "paint" type program with lines, boxes, etc.
gad	John Draper's Gadget tutorial program
gfxmem	Graphical memory usage display prog.
halfbrite	demonstrates "Extra-Half-Brite" mode, if you have it
hello	simple window demo
lattp	accessing the Motorola Fast Floating Point library from C
palette	Sample prog. to design color palettes.
trackdisk	Demonstrates use of the trackdisk driver.
requesters	John Draper's requester tutorial and example program.
speech	Sample speech demo program. Stripped down "speechtoy".
speechtoy	Another speech demo program.

Fred Fish Disk 2:

alib	Object module librarian.
cc	Unix-like frontend for Lattice C compiler.
dbug	Macro based C debugging package. Machine independent.
make	Subset of Unix make command.
make2	Another make subset command.
microemacs	Small version of emacs editor, with macros, no extensions
portar	Portable file archiver.
xrf	DECUS C cross reference utility.

Fred Fish Disk 3:

gothic	Gothic font banner printer.
roff	A "roff" type text formatter.
tf	A very fast text formatter
cforth	A highly portable forth implementation. Lots of goodies.
xlisp	Xlisp 1.4, not working correctly.

Fred Fish Disk 4:

banner	Prints horizontal banner
bgrep	A Boyer-Moore grep-like utility
bison	GNU Unix replacement 'yacc', not working.
bm	Another Boyer-Moore grep-like utility
grep	DECUS grep
kermit	simple portable Kermit with no connect mode.
MyCLI	Replacement CLI for the Amiga. V. 1.0
mandel	A Mandelbrot set program, by Robert French and RJ Mical

Fred Fish Disk 5

cons	Console device demo program with supporting macro routines.
freemap	Creates a visual diagram of free memory

For PDS orders, Please use form on page 128

input.dev — sample input handler, traps key or mouse events
joystick — Shows how to set up the gameport device as a joystick.
keyboard — demonstrates direct communications with the keyboard.
layers — Shows use of the layers library
mandelbrot — IFF Mandelbrot program
mouse — hooks up mouse to right joystick port
one.window — console window demo
parallel — Demonstrates access to the parallel port.
printer — opening and using the printer, does a screen dump, not working
printsupport — Printer support routines, not working.
proctest — sample process creation code, not working
region — demos split drawing regions
samplefont — sample font with info on creating your own
serial — Demos the serial port
singlePlayfield — Creates 320 x 200 playfield
speechtoy — latest version of cute speech demo
speech.demo — simplified version of speechtoy, with IO requests
text.demo — displays available fonts
timer — demos timer.device use
trackdisk — demos trakdisk driver

Fred Fish Disk 6:
compress — like Unix compress, a file squeezer
dadc — analog clock impersonator
microemacs — upgraded version of microemacs from disk 2
mult — removes multiple occuring lines in files
scales — demos using sound and audio functions
setparallel — Allows changing parallel port parameters
setserial — Allows changing serial port parameters.
sortc — quicksort based sort program, in C
stripc — Strips comments and extra whitespace from C source

Fred Fish Disk 7:
This disk contains the executables of the game Hack V 1.0.1.

Fred Fish Disk 8:
This disk contains the C source to Hack on disk 7.

Fred Fish Disk 9:
moire — Draws moire patterns in black and white
MVP-FORTH — Mountain View Press Forth, version 1.00.03A. A shareware version of FORTH from Fantasia Systems.
proff — a more powerful text formatting program
setface — Program to toggle interlace mode on and off.
skewb — a rubic's cube type demo
sparks — moving snake Graphics demo

Fred Fish Disk 10:
conquest — An interstellar adventure simulation game
dehex — convert a hex file to binary
filezap — Patch program for any type of file.
fixobj — Strip garbage off Xmodem transfered files.
iff — Routines to read and write iff format files.
ld — simple directory program
ls — Minimal UNIX ls, with Unix-style wildcarding, in C
sq,usq — file squeeze and unsqueeze
trek73 — Star Trek game
yachtc — Dice game.

Fred Fish Disk 11:
dpslide — slide show program for displaying IFF images with miscellaneous pictures

Fred Fish Disk 12:
amiga3d — Shows a rotating 3 dimensional solid "Amiga sign".
ArgoTerm — a terminal emulator program, written in assembler
arrow3d — Shows a rotating 3 dimensional wire frame arrow.
ld4 — directory listing program
IconExec
SetWindow — two progs for launching progs from Workbench, presently only works under CLI.
SetAlternate — Makes an icon show a second image when clicked once
StarTerm — terminal emulator, with ASCII Xmodem, dialer, more.

Fred Fish Disk 13:
A Bundle of Basic programs, including:

Jpad	toybox	ezspeak
mandlebrot	xmodem	3dsolids
addbook	algebra	ror
amgseq1	amiga-copy	band
bounce	box	brickout
canvas	card	circle
colorcircles	Copy	cubes1
cutpaste	date	dogstar
dragon	draw	dynamictriangle
Eliza	eterm	filibuster
fractal	fscape	gomoku
dart	haiku	hal9000
halley	hauntedM	hidden
join	loz	mandel
menu	minipaint	mouse
Othello	patch	pena
pinwheel	gbox	random-circles
Readme	rgb	rgbtest
Rord	sabotage	salestalk
shades	shapes	shuttle
sketchpad	spaceart	speakspeech
speecheasy	spell	sphere
spiral	striper	superpad
suprshr	talk	terminal
termtest	tom	topography
triangle	wheels	xenos
xmostriper		

(note: some programs are Abasic, most are Amigabasic, and some programs are presented in both languages)

Fred Fish Disk 14:
amiga3d — update of #12, includes C source to a full hidden surface removal and 3D graphics
beep — Source for a function that generates a beep sound
dex — extracts text from within C source files
dimensions — demonstrates N dimensional graphics
filezap — update of disk 10, a file patch utility
gfxmem — update of disk 1, graphic memory usage indicator
gi — converts IFF brush files to Image struct, in Ctext
pdterm — simple ANSI VT100 terminal emulator, in 80 x 25 screen
shell — simple Unix 'csh' style shell
termcap — mostly Unix compatible 'termcap' implementation.

Fred Fish Disk 15:
Blobs — graphics demo, like Unix 'worms'
Clock — simple digital clock program for the title bar
Dazzle — An eight-fold symmetry dazzler program. Really pretty!
Fish — double buffered sequence cycle animation of a fish
Monopoly — A really nice monopoly game written in AbasiC.
OkidataDump — Okidata ML92 driver and WorkBench screen dump program.
Polydraw — A drawing program written in AbasiC.
Polyfractals — A fractal program written in AbasiC.

Fred Fish Disk 16:
A complete copy of the latest developer IFF disk

Fred Fish Disk 17:
The NewTek Digi-View video digitizer HAM demo disk

Fred Fish Disk 18:
AmigaDsplay — dumb terminal program with bell, selectable fonts
Ash — Prerelease C Shell-like shell program, history, loops, etc.
Browser — wanders a file tree, displays files, all with the mouse
MC68010 — docs on upgrading your Amiga to use a MC68010
Multidim — rotate an N dimensional cube with a joystick
PigLatin — SAY command that talks in Pig Latin
Scrimper — Screen image printer
Xlisp1.6 — source, docs, and executable for a Lisp interpreter.

Fred Fish Disk 19:
BlackJack — text-oriented blackjack game
JayMinerSlides — Slides by Jay Miner, Amiga graphics chip designer, showing flowchart of the Amiga internals, in 640 x 400.
Keymap_Test — test program to test the key mapping routines
LockMon — Find unclosed file locks, for programs that don't clean up.

Fred Fish Disk 20:
AmigaToAtari — converts Amiga object code to Atari format
DiskSalv — program to recover files from a trashed AmigaDOS disk.
Hash — example of the AmigaDOS disk hashing function
Hd — Hex dump utility a la Computer Language magazine, April 86
MandelBrots — Mandelbrot contest winners
MultiTasking — Tutorial and examples for Exec level multitasking
Pack — strips whitespace from C source
PortHandler — sample Port-Handler program that performs. Shows BCPL environment clues.
Random — Random number generator in assembly, for C or assembler.
SetMouse2 — sets the mouse port to right or left
SpeechTerm — terminal Emulator with speech capabilities, XModem
TxEd — Demo editor from Microsmith's Charlie Heath

Fred Fish Disk 21
This is a copy of Thomas Wilcox's Mandelbrot Set Explorer disk. Very good!

Fred Fish Disk 22
This disk contains two new "strains" of microemacs.
Lemacs — version 3.6 by Daniel Lawrence. For Unix V7, BSD 4.2, Amiga, MS-DOS, VMS. Uses Amiga function keys, status line, execute, startup files, more.
Pemacs — By Andy Poggio. New features include <ALT> keys as Meta keys, mouse support, higher priority, backup files, word wrap, function keys.

Fred Fish Disk 23
Disk of source for MicroEmacs, several versions for most popular operating systems on micros and mainframes. For people who want to port MicroEmacs to their favorite machine.

Fred Fish Disk 24:
Conques — interstaller adventure simulation game
Csh — update to shell on Disk 14, with built in commands,named variables substitution.
Modula-2 — A pre-release version of the single pass Modula-2 compiler originally developed for Machtosh at ETHZ. This code was transmitted to the AMIGA and is executed on the AMIGA using a special loader. Binary only.

Fred Fish Disk 25
Graphic Hack — A graphic version of the game on disks 7 and 8. This is the graphics-oriented Hack game by John Toebes. Only the executable is present.

Fred Fish Disk 26
UnHunk — Processes the Amiga "hunk" loadfiles. Collect code, data, and bss hunks together, allows individual specificatio of code, data, and bss origins, and generates binary file with format reminiscent of Unix "a.out" format. The output file can be easily processed by a separate program to produce Motorola "S-records" suitable for downloading to PROM programmer. By Eric Black.
C-kermit — Port of the Kermit file transfer program and server.
Ps — Display and set process priorities
Archx — Yet another program for bundling up text files and mailing or posting them as a single file unit.

Fred Fish Disk 27
ABdemos — Amiga Basic demos from Caroly Scheppner.
NewConverIFD — creates .bmaps from fd files.
BitPlanes — finds addresses of and writes to bitplanes of the screen's bitmap.
AboutBmaps — A tutorial on creation and use of bmaps.
LoadILBM — loads and displays IFF ILBM pics.
LoadACBM — loads and displays ACBM pics.
ScreenPrint — creates a demo screen and dumps it to a graphic printer.
Disassem — Simple 68000 disassembler. Reads standard Amiga object files and disassembles the code sections. Data sections are dumped in hex. The actual disassember routines are set up to be callable from a user program so instructions in memory can be disassembled dynamically. By Bil Rogers.
DvorakKeymap — Example of a keymap structure for the Dvorak keyboard layout. Untested but included because assembly examples are few and far between. By Robert Burns of CA
Hypocycloids — Spirograph, from Feb. 84 Byte.
LinesDemo — Example of proportional gadgets to scroll a SuperBitMap.
MemExpansion — Schematics and directions for building your own homebrew 1 Mb memory expansion, by Michael Fellinger.
SafeMalloc — Program to debug 'malloc()' calls
ScienceDemos — Convert Julian to solar and sidereal time, stellar positions and radial velocity epoch calculations and Galilean satellite plotter. By David Eagle.

Fred Fish Disk 28
ABasic games by David Addison: Backgammon, Cribbage, Milestone, and Othello
Cpp — DECUS 'cpp' C preprocessor, and a modified 'cc' that knows about the 'cpp', for Manx C.
Shar — Unix-compatible shell archiver, for packing files for travel.
SuperBitMap — Example of using a ScrollLayer, syncing SuperBitMaps for printing, and creating dummy RastPorts.

Fred Fish Disk 29
AegisDraw Demo — Demo program without save and no docs.
Animator Demo — Player for the Aegis Animator files
Cc — Unix-like front-end for Manx C
Enough — Tests for existence of system resources, files, and devices
Rubik — Animated Rubik's cube program
StringLib
Vt100 — VT-100 terminal emulator with Kermit and Xmodem protocols

Fred Fish Disk 30
Several shareware programs. The authors request a donation if you find their program useful, so they can write more software.
BBS — an Amiga Basic BBS by Ewan Grantham
FineArt — Amiga art
FontEditor — edit fonts, by Tim Robinson
MenuEditor — Create menus, save them as C source, by David Pehrson
StarTerm3.0 — Very nice telecommunications by J. Nangano
(Fred Fish Disk#30 is free if requested when ordered with at least three other disks from the collection.)

Fred Fish Disk 31
Life — Life game, uses bitter to do 19.8 generations a second.
Mandelbrot — Version 3.0 of Robert French's program.
MxExample — Mutual exclusion gadget example.
RamSpeed — Measure relative RAM speed, chip and fast.
Set — Replacement for the Manx "set" command for environment variables,with improvements.
Tree — Draws a recursive tree, green leafy type, not flies.
TxEd — Crippled demo version of Microsmith's text editor, TxEd.
VDraw — Full-featured drawing program by Stephen Vermeulen.
Xicon — Invokes CLI scripts from icon
Ticon — Displays text files from an icon.

Fred Fish Disk 32
Address — Extended address book written in AmigaBasic.
Calendar — Calendar/diary program written in AmigaBasic.
DosPlus1 — First volume of CLI oriented tools for developers.
DosPlus2 — Second volume of CLI oriented tools for developers.

Executables only:
MacView — Views MacPaint pictures in Amiga low or high res, no sample pictures, by Scott Evernden.
Puzzle — Simulation of puzzle with moving square tiles.
ShowHAM — View HAM pictures from CLI
Solitaire — ABasiC games of Canfield and Klondike, from David Addison.
Spin3 — Graphics demo of spinning cubes, double-buffered example.
Sword — Sword of Fallen Angel text adventure game written in Amiga Basic.
Trails — Leaves a trail behind mouse, in Modula-2

Fred Fish Disk 33
3dstars — 3d version of the "stars" program below.
Bigmap — Low-level graphics example scrols bitmap with ScrollVPort
Dbuf.geis — Double-buffered animation example for BOBs and VSprites.
DiskMapper — Displays sector allocation of floppy disks.
MemView — View memory in real time, move with joystick.
Oing — Bouncing balls demo
Sproing — Oing, with sound effects.
ScreenDump — Dumps highest screen or window to the printer.
Sdb — Simple database program from a DECUS tape.
Stars — Star field demo, like Star Trek.
TermPlus — Terminal program with capture, library, function keys, Xmodem, CIS-B protocols.
Vt100 — Version 2.0 of Dave Wecker's VT-100 emulator, with scripts & function

Fred Fish Disk 34
Alnt — Support files for Gimpel's 'lint' syntax checker
Blink — PD 'slink' compaible linker,faster,better.
Browser — Updated to FF 18 'browser', in Manx, with scroll bars, bug fixes.
Btree — b-tree data structure examples
Btree2 — Another version of 'btree'
Calendar — Appointment calendar with alarm.
Less — File viewer, searching, position by percent, line number.
NewFonts — Set of 28 new Amiga fonts from Bil Fischer
Pr — Background print utility, style options, wildcards.
Requester — Deluxe Paint-type file requester, with sample.

Fred Fish Fred Fish Disk 35
ASendPacket — C example of making asynchronous I/O calls to a DOS handler, written by C-A
ConsoleWindow — C example of getting the Intuition pointer a CON: or RAW: window, for 1.2, by C-A.
DrUtil — Walk the directory tree, do CLI operations from menus
DrUtil2 — Another variant of DirUtil.
FileRequester — Lattice C file requester module, with demo driver, from Charlie Heath.
MacView — Views MacPaint pictures in Amiga low or high res, with sample pictures, by Scott Evernden.
Plop — Simple IFF reader program
PopCLI — Sidekick-style program invokes a new CLI, with automatic screen blanking.
QuickCopy — Devenport disk copiers duplicate copy-protected disks.
ScrollPf — Dual playfield example, from C-A, shows 400 x 300 x 2 bit plane playfield on a 320 x 200 x 2 plane deep playfield.
SendPacket — General purpose subroutine to send AmigaDos packets.
SpriteMaker — Sprite editor, can save work as C data structure. Shareware by Ray Larson.
Tracker — Converts any disk into files, for electronic transmission. Preserves entire file structure. Shareware by Brad Wilson.
TriClops 3-D — space invasion game, formerly commercial, now public domain. From Geodesic Publications.
Tsize — Print total size of all files in subdirectories.
Unifdef — C preprocessor to remove given #ifdef'd sections of a file, leaving the rest alone. By Dave Yost
Vtest — VT-100 emulation test program. Requires a Unix system.

Fred Fish Disk 36
Acp — Unix-like 'cp' copy program
Clock — Updated version of clock on disk 15.
Csh — Manx 'csh'-like CLI, history, variables, etc.
DietAid — Diet planning aid organizes recipes, calories
Echo — Improved 'echo' command with color, cursor addressing
FixHunk — Fixs programs to let them run in external memory.
Fm — Maps the sectors a file uses on the disk.
KickBench — Docs, program to make a single disk that works like a Kickstart and Workbench.
Lex — Computes Fog, Flesch, and Kincaid readability of text files.
TunnelVision — David Addison ABasic 3D maze perspective game.
Vc — Visicalc-like spreadsheet calculator program.
Vt100 — Version 2.2 of Dave Wecker's telecom program
YaBoing — Oing! style game program shows sprite collision detects

Fred Fish Disk 37
This disk is a port of Timothy Budd's Little Smalltalk system, done by Bill Kinnersley at Washington State University.

Fred Fish Disk 38
CSquared — Sep 86 Sci American, Circle Squared algorithm
FixObj — Strips garbage off Xmodem transfered object files
Handler — AmigaDOS handler (device) example from C-A
Hp-10c — Mimics a HP-10C calculator, written in Modula-2
IFFEncode — Saves the screen as an IFF file
IffDump — Dumps info about an IFF file
Jsh — BDS C-like CLI shell
NewStat — STATUS-like program, shows priority, processes
Reversi — Game of Reversi, version 6.1

UUdecode — Translate binary files to text, Unix-like programs
Vdraw — Drawing program, version 1.14
VoiceFiler — DX MIDI synthesizer voice filer program
Window — Example of creating a DOS window on a custom screen

Fred Fish Disk 39

AnsiEcho — 'echo', 'touch', 'list', 'cls' written in assembler.
Display — Displays HAM images from a ray-tracing program, with example pictures.
Driver — Example device driver source, acts like RAM: disk
Xlisp — XLisp 1.7, executable only

Fred Fish Disk 40

Ahost — Terminal emulator with Xmodem, Kermit and CIS B protocols, function keys, scripts, RLE graphics and conference mode.
AmigaMonitor — Dynamically displays the machine state, such as open files, active tasks, resources, device states, interrupts, libraries, ports, etc.
Arc — Popular file compression system, the standard for transiting files
AreaCode — Program that decodes area codes into state and locality.
Blink — 'alink' replacement linker, version 6.5
Cosmo — An 'asteriods' clone.
Dg210 — Data General D-210 Terminal emulator
DirUtil — Windowed DOS interface program, V 1.4
DOSHelper — Windowed AmigaDOS CLI help program
PagePrint — Prints text files with headers, page breaks, line numbers
PopCLI — Starts a new CLI with a single keystroke, from any program, With a screen-saver feature. Vessure2, with
SpriteEd — Sprite Editor edits two sprites at a time
X-Spell — Spelling checker allows edits to files

Fred Fish Disk 41

AmigaVenture — Create your own text adventure programs in AmigaBasic.
Csh — Version 2.03 of Dillon's C sh-like shell. Executable only
Dbug — Macro based C debugging[illegible]#2
DualPlayField — example from CBM, update to Intuition manual
GetFile — Heath's file requester, with source
LatXref — Cross reference of Lattice 3.10 header files
Lines — Line drawing demo program
SetFont — Changes font used in a CLI window
Vt100 — Version 2.3 of the VT-100 terminal program.

Fred Fish Disk 42

This disk contains an Amiga version of MicroGNUEmacs.

Fred Fish Disk 43

BasicBoing — AmigaBasic program demos page flipping of a 3D cube
Bom — Demo copy of B.E.S.T. Business Management System.
BosList — A list of Amiga Bulletin Board Systems
Cc — C compiler frontends for Manx and Lattice C
Copper — A hardware copper list disassembler
InstIFF — Converts Instruments demo sounds to IFF sampled sounds
PopColours — Adjust RGB colors of any screen
SpriteClock — Simple clock is displayed on a sprite above all screens
ST Emulator — Non-serious Atari ST emulator
WBrun — Lets Workbench programs be run from the CLI
Wild — Two Unix shell style wild card matching routines

Fred Fish Disk 44

Icons — Miscellaneous icons
NewIFF — New IFF material from CBM for sampled voice and music files
RayTracePics — The famous ray-tracing pictures, from FF#39, now converted to IFF HAM format for *much* faster viewing.
ViewILBM — Displays normal and HAM ILBM files

Fred Fish Disk 45

Clue — Clue board game
Make — Another 'make', with more features
Pictures — Miscellaneous pictures
Update — Updates an older disk with newer files from another disk
Wherels — Searches a disk for files of given name

Fred Fish Disk 46

Asm — Shareware 68010 macro assembler, ROM Kernal Manual compatible
CheckModem — 'execute' file program detects presence of modem
Egad — Gadget editor from the Programmers Network
Jive — Transforms a file from English to Jive.
MyJib — A binary only copy of Ma tt's alternate runtime library. Author: Matt Dillon
ProffMacros — Subset Berkeley 'ms' and 'mm' macros for 'proff'
ValSpeak — Transforms a file from English to Valley Speak.

Fred Fish Disk 47

3D-Arm — Simulation of a robotic a rm, very good graphics, teaching tool, including C source.
Juggler — Eric Graham's stunning HAM animation of a robot juggler
VT-100 — Version 2.4 of Dave Wecker's terminal emulator, with Xmodem and Kermit file transfer protocols

Fred Fish Disk 48

Bru — Alpha version of a hard disk file archiver
Comm — Version 1.30 of a terminal emulator with phone directories
Csh — Version 2.04 of Matt Dillon's Unix 'csh'-like CLI replacement, including Lattice and Manx C source
Diskperf — Disk benchmark program for Unix and Amiga
Du — Computes disk storage of a file or directory
MemWatch — Program to watch for programs that trash low memory. It attempts to repair the damage, and puts up a requester to inform you of the damage. From the Software Distillery.
Profiler — A realtime execution profiler for Manx C programs. Includes C source.

Fred Fish Disk 49

Cydoids — Update of electronic spirograph from disk 27
DirUtil — Enhanced version of DirUtil from disk 35
MultDef — Scans a set of object modules and libraries searching for multiply defined symbols
MyUpdate — Disk update utility with options for stripping comments from C header files, and interactive verification of the updating process
Plot — Computes and displays 3 dimensional functions in hires
Polygon — Moire type pattern generator with color cycling
QMouse — Queries whether a mouse button is pressed. This can give a return code that can customize a startup-sequence based on whether a mouse button was pressed.
Touch — Example of setting the datestamp on a file, using a technique from Commodore-Amiga
Trees — More extensive version of the trees program on Disk 31

Fred Fish Disk 50

Asm — Version 1.1 of a shareware 68000 macro assembler, compatible with the Metacomco assembler. This includes an example startup module and more Motorola mneumonics.
BreakOut — A brick breakout game, uses 3-D glasses
DiskZap — Version 1.1 of a program to edit disks and binary files
FirstSilicon — A smart CLI replacement with full editing and recall of previous commands
Missile — A Missile Command-type game, with sound, in assembler
PerfectSound — Sound editor for a low-cost sound digitizer
Sizzlers — Graphics demos
UnixArc — Version of 'arc' for Unix System V machines, in C
Wombat — Version 3.01 of Dave Warker's terminal emulator

Fred Fish Disk 51

Bison — GNU for Unix 'yacc', working update to disk 4 version
Compress — Update to the file compression program on Disk 6
Cos — "Wheel of Fortune"-type game in AmigaBasic
DfSeed — Unix-like 'dif' and 'sseq' for finding the differences between two files, and then recreating the other, given one file, and the list of differences.
Sq, Usq — Portable versions of the CP/M squeeze and unsqueeze

Fred Fish Disk 52

Assign — Replacement for AmigaDOS 'assign' command in C
Fractal — Makes random fractal terrains
Poly, HAMPoly — Workbench-type demos for making polygons in lores and HAM
MxGads — Example of mutual exclusion gadgets wth GadgetText
Tek4010 — Tektronix 4010 terminal emulator
VDraw — Versions 1.16 and 1.19 of a Deluxe Paint-like drawing program

Fred Fish Disk 53

Animations — Demo animations with player program for Aegis Animator
ARCre — Creates rename scripts for files with long names, so they can be easily 'arc'ed and un'arc'ed.
ARP — Preliminary AmigaDOS replacements for 'break', 'cd', 'chmod', 'echo', 'filenote' and 'makedir'
Compiler — Not fully ported to the Amiga, this is a 68000 C compiler. It wil produce simple assembly language output, but needs a lot of work.
Spreadsheet — Update with source of the 'vc' spreadsheet on disk 36
TarSplit — Port of program to split Unix 'tar' archives
UUencode — Utilities to encode and decode binary files for ASCII transmission, expanding them by 35 percent Paint-like drawing program

Fred Fish Disk 53

Animations — Demo animations with player program for Aegis Animator
ARCre — Creates rename scripts for files with long names, so they can be easily 'arc'ed and un'arc'ed.
ARP — Preliminary AmigaDOS replacements for 'break', 'cd', 'chmod', 'echo', 'filenote' and 'makedir'
Compiler — Not fully ported to the Amiga, this is a 68000 C compiler. It wil produce simple assembly language output, but needs a lot of work.
Spreadsheet — Update with source of the 'vc' spreadsheet on disk 36
TarSplit — Port of program to split Unix 'tar'archives
UUencode — Utilities to encode and decode binary files for ASCII transmission, expanding them by 35 percent

Fred Fish Disk 54

Hanoi — Solves Towers of Hanoi Problem in it's own Workbench window. by Ali Ozer
ISpell — Port of a Unix screen oriented, interactive spelling checker. (Expansion RAM required) by Pace Willisson
Ihg — A Screen of lots of bouncing little windows by Leo 'Bols Ewhac' Schwab
Lav — Displays number of tasks in run queue, averaged over last 1, 5, and 15 minute periods. by William Rucklidge
MIDITools — Programs to play/record through the MIDI I/F. by Fred Cassirer
MoreRows — Program to make the Work Bench Screen larger than normal. by Neil Katin and Jim Mackraz
Tilt — Program to make your Amiga look like it didn't pass vibration testing. by Leo " Bols Ewhac' Schwab

Fred Fish Disk 55

Csh — V2.05 of Matt Dillon's csh like shell (Modified for Manx C). by Matt Dillon, Modified by Steve Drew
NewStartups — New C Startup modules:
AStartup.asm — with 1.2 fixes and better quote handling.
TWStartup.asm — opens a stdio window, using user specs. by Commodore, posted to BIX by Carolyn Schepper
Palette — Change another program's screen colors. by Carolyn Schepper
PipeDevice — Allows the standard output of one process to be fed to the standard input of another. by Matt Dillon
ScreenSave — Save a normal or HAM mode screen as an IFF file. by Carolyn Schepper
ShanghaiDemo — Demo of the Activision game Shanghai.
SoundExample — A double buffered sound example for Manx C. by Jim Goodnow
Vsprites — A working vsprite example. by Eric Cotton
Vt100 — V2.5 of Dave's Vt100 terminal emulator with kermit and xmodem. by Dave Wecker

Fred Fish 56

ClipBoard — Clipboard device interface routines, to provide a standard interface. by Andy Finkle
ConPackets — Demos the use of DOS Packets, ConUnit, etc. by Carolyn Schepper
GetDisks — Program to find all available disk device names and return them as an exec list. by Philip Lindsay
GetVolume — Program to get volume name of the volume that a given file resides on. by Chuck McManis
Icon2C — Reads an icon file and writes out a fragment of C code with the icon data structures. by Carolyn Schepper
MergeMem — Program to merge the MemList entries of sequentially configured RAM boards. by Carolyn Schepper
mCAD — An object oriented drawing program, V1.1 by Tim Mooney

Fred Fish 57

Replaced by FF97 Due to Copyright problems

Fred Fish 58

ASDG-rrd — Extremely useful shareware recoverable ram disk. by Perry Kivolowitz
BigView — Displays any IFF picture, independent of the physical display size, using hardware scroll. by John Hodgson
EGraph — Reads pairs of x and y value from a list of files and draws a formatted graph. by Laurence Turner
HyperBase — Shareware data management system. V1.5
MemClear — Walks through the free memory lists, zeroing free memory along the way. by John Hodgson
NewZAP — A third-generation multi-purpose file sector editing utility. V3.0 by John Hodgeson
RainBow — A Mauraider-Style rainbow generator. by John Hodgson
SMUSPlayers — Two SMUS plays, to play SMUS IFF music formatted files. by John Hodgson
View — A tiny ILBM viewer by John Hodgson
WBdump — JX-80 optimized workbench printer that does not use DumpRPort. by John Hodgson

Fred Fish Disk 59

Browser — Update to browser program on disks 18 and 34. S-E
Browser2 — Another different browser program. E
Clock — Clock program with fonts, colors. E
Dme — Dillon text editor V1.22 for programmers. E-D
DropCloth — Puts a pattern on the Workbench backdrop. E-D
DropShadow — Puts shadows on Workbench windows. E-D
FixWB — Similar to DropCloth, but doesn't work yet. S-D
mCAD — Object-oriented drawing program, version 1.2.2. Much improved over disk 56.
Robotroff — Demo of animated pointers on Workbench. S-E-D
Supermort — General compounding/amortization loan calculator. E-D

Fred Fish Disk 60

Various shareware and freeware programs
Bitz — Memory resident file viewer. Very fast. E-D
BitzFonts — Makes text output faster. E-D
HandShake — Terminal emulator with VT52/VT100/VT102support. E-D
Med — Mouse-driven text editor version 2.1. E-D
PrDrvGen — Generates printer drivers, version 1.1. Source available from author. E-D
Show — Slideshow-like IFF viewer, V2.1. E-D
Uedit — Customizable text editor V2.0. E-D
Ueturbo — Example Uedit setup macros. S-E-D

Fred Fish Disk 61

ATPatch — Patches Transformer to work under AmigaDOS 1.2. S-E-D
FillDisk — Writes zeroes to free blocks on a disk for security. S-E-D
LPatch — Patch for programs that abort when loading under AmigaDOS 1.2. S-E-D
MicroEmacs — Conroy MicroEmacs V3.8b, newer than disk 22. S-E-D
PearlFont — Like Topaz, but rounded edges.
Terrain — Generates fractal scenery. S-E-D
VSprites — Makes 28 Vsprites, from P&E Book.

Fred Fish Disk 62

This is a port of the Unix game 'Hack', by the Software Distillery, version 1.0.3D.

Fred Fish Disk 63

This is a port of the Unix game 'Larn', by the Software Distillery, version 12.0B.

Fred Fish Disk 64

This is an offical IFF specification disk from Commodore, an update to disk 16.

Fred Fish Disk 65

Awk — Unix text processor, like 'awk'. Doesn't work, but source is included. S-E-D.
MWB — Example of rerouting Workbench window open calls to another custom screen. Version 1.01, S-E-D
CloseWB — Example for closing a custom Workbench screen. S-E-D
Cookie — Generates one-line fortune-cookie aphorisms. S-E-D
JTime — Build-your-own mouse port clock.
MenuBuilder — Creates C source files for menus, based on text descriptions. S-E-D.
NewPackets — CBM tutorial on new packets and structures in AmigaDos 1.2.
PascalToC — Pascal to C translator, not so great. S-E-D
Prep — 'ratfor'-like FORTRAN preprocessor. S-E-D
RunBack — Starts programs from CLI, allowing CLI window to close. E-D
SunMouse — This program automatically clicks in windows when the mouse is moved over them. Version 1.0, E-D

Fred Fish Disk 66

AmScsi — Preliminary plans for a SCSI disk controller board.
Asm68k — Macro assembler, version 1.0.1. E-D
Assigned — Example for avoiding DOS insert-disk requester, by scanning the list of 'assign'ed names. S-E-D
Dk — Pretends to eat away at CLI window. S-E-D
Flip — Flips whole screen as a joke. S-E-D
Foogol — Foogol cross-compiler generates VAX assembly code. S-E-D
Free — Prints amount of free space on all drives. S-E-D
MallocTest — malloc/free memory test program. S-E-D
Melt — Pretends to melt the screen. S-E-D
Nart — Graphic flying string demo. S-E-D
Purty — Easy way to set printer attributes from Workbench. E-D
RayTracer — Simple ray tracing program. E-D
SendPackets — Updated CBM examples of packet routines on disk 35. S-E-D
SnapShot — Memory resident screen dump. E-D
TagBBS — Shareware BBS system, version 1.02.

Fred Fish Disk 67

AmCat — Shareware disk cataloging program.
AmigaSpell — Shareware Intuition spelling checker, V2.0. E-D
Bouncer — 3-D bouncing ball written in MultiForth, S-E-D
Comm — Terminal program version 1.33, E
Dux5 — Another version of DirUtil. S-E-D
HexCalc — Hex, octal, & decimal calculator. E-D
Icons — Various big and alternate image icons.
Mandala — Mandala graphics and sound. E
PersMail — Demo shareware personal file manager.
RSLClock — Menu bar clock version 1.3. E-D
RTCubes — Graphics demo of 3D cubes. E-D
Wheel — "Wheel of Fortune"-type game, in AmigaBasic

Fred Fish Disk 68

This is version MG 1b of the MicroGNUEmacs. Source and executable areincluded, as well as source for other computers besides the Amiga.

Fred Fish 69

Asm68k — Macro assembler, v1.0.3, E-D
BitLab — Bitter exploring program, in C, S-E-D
Conman — Replacement console device handler adds editing and history to any application that uses CON:, v0.9, E-D
Console — Replacement console routines, in C, S-E-D
Dk — Decays the screen bit by bit, update to disk 66, in Modula-2, S-E-D
Frags — Displays memory fragmentatiby listing the size of free memory blocks, in C, S-E-D
IconType — Change the type of an icon, in C, S-E-D
Make — 'make' in Manx C, S-E-D
MonProc — Monitors processes for packet activity, in C,S-ED
MouseClock — Turns mouse pointer into a digital clock,in C,SED
Sb — Browses system structures, from Transactor magazine, v1.0, in C, S-E-D
Spew — Generates 'National Enquirer'-type headlines from rules file. In C,S-E-D
Spool — Three programs to demonstrate multitasking and spooling in a printer spooler. In C, v1.2, S-E-D
Wc — Counts words a/a Unix 'wc', but faster, in C,S-ED

Fred Fish 70

This is a disk of shareware programs.
AmigaMonitor — Explores state of the system, v1.13
Arc — Standard file compressor and librarian, v0.23, a port of MS-DOS v5.0. E-D
BlackBook — Phone book program.
DoTil — Intuition-driven file manipulator program,v2.0.
GravityWars — Game of planets, ships and black holes,v1.03.
Jobs — Alternate user interface to CLI and WB, v2.1.

For PDS orders, Please use form on page 128

Lens	Magnifies area around mouse, shows it in a window, v1.0.
Life-3d	3D version of the classic cellular-automaton game, v1.2.
Logo	Logo language interpreter
SetKey	Demo keymap editor, v1.0
Vpg	Makes displays for aligning video monitors, v1.0.

Fred Fish 71

AirFoil	Makes airfoils using the Joukowski transformation, in C, S-E-D
Amiga Basic	Miscellaneous programs including 3D plot program, a kaleidoscope, C-A logo drawing program file comparison utility string search program, S-E-D
Blocks	A variation of "lines", but with variable color blocks. E-D
Comm	Great terminal program, v1.34, E-D
DiskX	Utility for exploring file system.E-D
Fpic	Simple image processing program that operates on IFF pictures, with several filters, merging images, E-D
IconMk	Makes icons for files , v1.2a, E-D
Icons	New icons
NewFonts	Two new fonts; 'shalt18', an electronic ciruit element font, and 'ibm5', a PC-like font.
PetCLI	An AmigaBASIC CLI shell program.
PWDemo	Demo of the commercial product PowerWindows,v1.2. It aids creation of custom windows, menus, and gadgets, giving C or assembly source. E-D
Rot	Creates and animates 3-D objects, v0.5, E-D
TimeSet	Sets time from Workbench, E-D

Fred Fish 72

This is a disk of IFF pictures.

Fred Fish 73

Add	Customizes existing program menus with Amiga-key shortcuts. Also includes 'until', which waits until a given window is created. Shareware, in C, S-E-D.
AutoIconOpen	Fools WB into thinking mouse has double-clicked icons. In C, S-E-D
Dio	Generic Exec device interface code for opening libraries, getting multiple I/O channels, asynchronous operations, etc. In C, S-E-D.
Dissolve	Slowly displays IFF files, a/a Nov 86 Dr. Dobb's program. In C, S-E-D
DTerm	Flexible, reprogrammable terminal program v1.10, E-D
Expose	Re-arranges windows so that at least one pixel of menu bar gadgets are exposed. In C, S-E-D.
Lit	Scans a text file, converts to C-style printable strings.C,v2.0, S-E-D
Lmv	"Long Movie", program views series of IFF picts in quick succession, upto 19 fps. Shareware, E-D
MouseOff	Mouse pointer disappears after ten seconds of non-use. In C, S-E-D
ParOut	Examples of controlling parallel port with resources instead of the PAR: device.In C,S-E-D
PenPalFont	Child-like font.
RunBackGround	Similar to RunBack on disk 66, runs program from the CLI allowing the CLI window to close. In C,S-E-D
SnapShot	Screendump utility,update FF 66.E-D
TypeAndTell	Example installs a device handler before Intuition, and speaks each key as it is pressed. In C and assembler, S-E-D
Xplor	Prints info about system lists, in assembler,S-E-D

Fred Fish Disk 74

Cled	Edits and recalls CLI commands, v1.3, E-D
Control	Intercepts graphic printer dump calls and accesses color map, width,and screen resolution. C,S-E-D
Dme	Simple WYSIWYG text editor for programmers,v1.25. Update of FF 59.E-D
DropShadow	Workbench dropshadows, v2.0. Update to disk 59. E-D
Funds	AmigaBASIC program tracks mutual or stock p-D
Less	Text viewing program, like Unix 'more', v1.1, update to disk 34. S-E-D
Makemake	Scans C source files and constructs a vanilla 'makefile' in the current directory. S-E-D
mCAD	Object-oriented drawing prog. v1.2.4, update to FF 59.Shareware, E-D
Random	Simple random number generator in C. S-E-D
TDebug	Monitors devices by intercepting Exec SendIO() and DoIO() vectors, in C, v1.0, S-E-D
Units	Converts measurements in different units, includes "chart" option, in C, S-E-D
XCopy	Replacement for AmigaDOS 'copy', doesn't change the date, uses Unix wildcards. E-D

Fred Fish Disk 75

Bezier	Play with Bezier curves points and granularity, S-E-D
BSplines	Play with b-splines, as above, S-E-D
Comm	C source for Comm terminal program v1.34. S-E-D
Copy	Replacement 'copy' command v1.0, preserves date, in C, S-E-D
Diff	Simple 'diff' in C, S-E-D
DuM2	Another DirUtil in Modula-2, v1.5, S-E-D
Eless	Fast 'dir' program in C, S-E-D
Fd	Faster 'eless' in C, S-E-D
HardCopy	Sends a transcript of a CLI session to a file, in C, S-E-D
MouseOff	Update to disk 73, turns off mouse pointer, S-E-D
SetFont	Changes the font in a Workbench screen, v2.0, S-E-D
SpeedDir	Another fast 'dir', in assembler, S-E-D

Fred Fish Disk 76 & 77

These are disks 1 and 2 of Chris Gray's Draco distribution for the Amiga. Draco is a compiled, structured language reminiscent of both C and Pascal. A full interface to AmigaDOS and Intuition is supplied. Be sure to get both disk 76 and 77.

Fred Fish Disk 78

Cycles	Cycle game like "Tron", v1.0, E-D
EOMS	Experts Only Mercenary Simulator game, E-D
MandelVroom	Mandelbrot generator with enhanced palette controls, fixed/floating point, presets, v1.50, in Manx C, S-E-D

Fred Fish Disk 79

AsmTools	CLI tools in assembler: echo, loadit, mounted, setdate, why; S-E-D
AssignDev	Give devices multiple names, in C, S-E-D
AuxHandler	Example of a dos handler that allows use of a CLI via the serial port. Includes source. Author: Steve Drew
Cmd	Redirects printer output to a file, in C, S-E-D
Info	AmigaDOS 'info' replacement, in C and assembler, S-E-D
Kill	Removes a task and its resources, in C, S-E-D
M2Error	Displays errors from TDI Modula-2 compiles, S-E-D
MonProc	Update to process packet program from disk 69, in C, S-E-D
Mounted	Program for testing if a drive is present, in a script In C, S-E-D
Nro	Another 'roff'-style text formatter, in C, S-E-D
ParTask	Finds parent task, in C, S-E-D
QueryAny	For scripts, asks a question, accepts Y/N, gives return code. In assembler, S-E-D
ScnSizer	Resets preferences settings for screen size, in C, S-E-D
SharedLib	Example of a shared library, in C and assembler, S-E-D
Task	Simple CreateTask() example in C, S-E-D
Uw	Unix Windows client v1.0, in C, S-E-D
Who	Lists tasks on ready and wait queues, in C, S-E-D

Fred Fish Disk 80 (see Fred Fish 90)

Fred Fish 80 has been withdrawn due to copyright problems.

Fred Fish Disk 81

Asm68k	V1.1.0 of a macro assembler
AutoFacc	Shrinks the FACC window and moves it to the back
Brushes	53 custom IFF brushes of electronic symbols
CheckIFF	Checks structure of an IFF file CledV1.4 update to disk 74 of a simple command line editor
Conman	Replaces console handler to add editing and history to many programs
Fonts	Miscellaneous fonts
Icon	V6.0 of the Icon programming language
KeyLock	Freezes the keyboard and mouse until pass word entered.
ScatDisplay	hack created from "fing"
Smush	Smushes an IFF file.
Target	Each mouse dlikc becomes a gunshot

Fred Fish Disk 82

AdventureA	port of the classic Crowther and Woods Adventure game
AmicTerm	V0.50 of a telecommunications program, with scripts, redial, beeps, enhanced file requester
D2D-Demo	Demo version of Disk-2-Disk from Central Coast Software
DX-Synth	Voice filer program for Yamaha DX series synthesizers, update to disk 38
DiskMan	V1.0 of another DirUtil program
Icons	Miscellaneous new icons
Pan!	Universal MIDI patch panel, v1.2
Rocket	Another Workbench hack, plays Lunar Lander
Sand	Game of sands following your pointer.

Fred Fish Disk 83

This disk contains a demo version of TeX from N Squared. It is limited to small files, and the previewer can only display ten pages or less, and only a small number of fonts are provided.

Fred Fish Disk 84

AudioTools	Programs from Rob Peck's July/August Amiga World article
BitLab	Bitter experimentation program, V1.2, update to disk 69
Ed	Simple editor, similar to Unix 'ed', based on the editor in Software Tools.
GravityWars	Game of planets, ships and black holes, v1.04, update to disk 70.
HunkPad	Adds legal padding to executables for Xmodem transmission.
PipeHandler	An AmigaDOS pipe device which supports named pipes and taps. V1.2
PopCLI	V3.0 of a hot-key to invoke a CLI window, with screen blanker, update to disk 40.
Requester	Update to disk 34 of a file requester similar to DPaint.
ScottDevice	V33.1 of a 'mount'able MicroForge SCSI driver.
Viacom	Another Schwab hack, makes TV-like static on screen.

Fred Fish Disk 85

Csh	V2.06 of Dillon's 'csh'-like shell
FileReq	Source to wildcard file requester
Hide	Hides expansion memory from programs
ImageTools	Shareware tools to manipulation IFF images
LowMem	ServerShared library to aid in low memory situations
Plot5	A star plotting program with source.
RawID	Example of setting raw mode on standard input
Rocket	Lunary Lander for Workbench, with source.
VMore	'more'-like text viewing utility, v1.0 with source.
Vnews	Simple Unix news reader.

Fred Fish Disk 86

AutoPoint	Auto-selects window under the mouse pointer, with screensaver.
ClickToFront	Double-clicks in window brings it to front, v1.1, S-E-D
Cmd	V3.0 of a tool to redirect printer output to a file.
FileIISG-Demo	Demo of Softwood File IIsg, a database manager with sound and graphics.

Fred Fish Disk 87

AdvSys	Adventure system from Byte May 1987, v1.2 E-D
AutoIconOpen	Fools Workbench to open disk icons, V1.2 update to disk 73, S-E-D
Claz	Converts IFF files to PostScript, V2.0, S-E-D
Commodi	tesMackraz's Commodities Exchange, an exec library to manage the input handler, v0.4
Diff	Update to disk 75 of Unix-like 'diff', S-E-D
Dme	V1.27 of Dillon's text editor, update to disk 74 , E-D
DropShadow	V2.0 of program that puts shadows on Workbench, S-E-D
Elib	Shared library example in Manx C.
ID-Handler	An AmigaDOS device handler that generates unique identifiers, V1.0, S-E-D
Install	Alternate AmigaDOS 'install' programs, S-E-D
MemWatch	Waits for low memory trashing, V2.0, S-E-D
MovePointer	Moves pointer to given location, S-E-D
MoveWindow	Move window to given location, S-E-D
MunchingSq	Munching Squares hack, S-E-D
PalTest	Example shows test to see if this is a PAL machine, S-E-D
Sc	Generates random scenery, S-E-D
Tek4695	Tek4695 printer driver
WBDualPF	Example of dual-playfield screen, update to disk 41, S-E-D
WarpText	Fast text rendering routines, S-E-D
Yaiff	rExample IFF reader, S-E-D
Zoo	A file archiver like "arc", v1.42A, E-D

Fred Fish Disk 88 (see Fred Fish 89)

Fred Fish Disk 88 has been removed due to copyright problems

Fred Fish Disk 89 (replaces Fred Fish 80)

DirMaster	Disk catalogue program, V1.0a, E-D
FuncKey	Shareware function key editor, V1.01, E-D
MFF-Demo	Demo of MicroFiche Filer database program
ScreenShift	Adjust screen position like Preferences, S-E-D
Snake	Bouncing squiggly lines demo, S-E-D
AutoEnguirer	screen contraption requester improvement S-E-D
DemoLition	Display Hack S-E-D

Fred Fish Disk 90 (replaces Fred Fish 80)

AmiGazer	Night sky viewer of 1573 stars, set date, time, day. E-D
CardFile	AmigaBasic card file study aid. E-D
Conman	Console handler replacement gives line editing and history to most progs, v0.98,E-D
MandelVroom	Slight update to disk 78 Mandelbrot program, E-D
NewDemos	Replacements for lines and boxes demos that take less CPU time, E-D
Othello	Game of Othello, E-D
PrnText	Displays text files with gadgets, speech, IFF display, v1.2, E-D
PrtDrvGen	Automatic printer driver generator, v2.2b, E-D
RainBench	Cycles colors of Workbench backdrop or text. E-D
ShortCut	Makes single-key shortcuts for entering commonly typed CLI commands, as well as custom macros. E-D
ShowPrint	Displays and prints IFF pictures of all sizes, and controls printer output styles, v2.0 E-D
Sizzlers	Graphics demos, v1.7.0, E-D
Timer	Small Workbench timer counts time and $/ minute, E-D
Tools	Inovatronics tools: a memory editor, memory disassembler, ASCII chart, and calculator. E

Fred Fish Disk 91

Adventure Definition Language (ADL) a superset of an older language called DDL by Michael Urban, Chris Kostanick, Michael Stein, Bruce Adler, and Warren Usui. ADL enhancements by Ross Cunniff. Included are sources to the ADL compiler, interpreter, and debugger. Binaries combined by Ross with Lattice 3.03. CLI enviornment only. Documentation is available from the authors.

Fred Fish Disk 92

As6502	portable 6502 assembler, including C source. by J. Van Ornum; Amiga port by Joel Swank
Bawk	Text processor update from FF65 Inspired by UNIX awk. Bawk searches files for patterns, performs actions based on patterns. By Bob Brodt; Amiga port by Johan Widen
HunkPad	update of FF84 version. by J. Hamilton, pads an object file to a multiple of 128 bytes for better xmodem transfer. S E
Less	Like Unix "more", better, version 1.2 update of FF74. Scrolls Back and f orward. S E by Mark Nudelman, Amiga port by Bob Leivian.
Ndir	Library that implements the 4BSD unix dir access routines by MikeMeyer. S
Parse	Recursive descent expression parcer, computes, and prints expressions. includes transcendental function support. c Source included. by John Olsen
Shar	Two programs to pack and unpack shell archives (traditional Usenet bundling of multiple text files for posting or electronic mail). includes C source. by Fabbian G. Dufoe
SmallLib	8 times smaller Amiga.lib replacement, binary only. by Bryce Nesbitt
UUencode	Encode/decode binary files for e mail or text-only methods. Update of FF53, includes checksum technique, compatible with older versions, plus transparent to older versions options. By Mark Horton, modified by Alan Rosenthal and Bryce Nesbitt.

Fred Fish Disk 93

Dme	Version 1.27 WYSIWYG programmer editor. Not a word processor. Includes key mapping, fast scrolling, title-line statistics, multiple windows, ability to iconify windows. Update of FF87, i ncludes source code. by Matt Dillon
MicroEmacs	Version 3.8i, update to FF61 includes source. Orig by Dave Conroy multiple modifications by Daniel Lawrence

Fred Fish Disk 94

AudioTools	Demo programs from Rob Peck's July/ August issue of AmigaWorld on accessing the audio device. Version 2, update of FF84. Includes source. by Rob Peck
ClickUpFront	Similar in function to ClickToFront program (FF86), bring windows to the front by clicking on any part of them. V 1.0. by Davide Cervone SE
HeliosMouse	Automatically activate a window simply by moving the mouse pointer into the window. V 1.0. Includes source. By Davide Cervone
IFF2Ps	Convert any IFF file to postscript for printing or viewing on a postscript compatible device. Version 1.2, by William Mason and Sam Paolucci E
ModulaTools	Various Modula 2 programming routines. by Jerry Mack
Terrain3d	Pseudo-random 3d relief scenery generator, update of "sc", FF87, by Chris Gray, 3d by Howard Hull

Fred Fish Disk 95

Cmd	redirects the serial.device or parallel. device output to a file. Usefull for capturing print jobs for debugging or "offline" printing. V4 By C Schepner SE
CygnusEdDemo	Demo of CygnusSoft's CygnusEd editor, a multipleflle, multiple feature editor. also includes demo 3.0 of MandFXP. by CygnusSoft Software E
Gomf	"Get Outa My Face" makes the Guru go away and allows the user to clean-up and shutdown more cleanly. V 1.0, by Christan Johnsen E
Journal	records sequence of mouse and keyboard events, stores them in a file for future playback. Good for demos or documenting bugs. E. by D. Cervone
MergeMem	attempts merging of MemList entries of sequentially configured ram boards. When successful, allows allocating a section of memory which spans both boards. V 2, update of FF56. by Carolyn Schepner SE
PrinterStealer	Asimilar to "Cmd", allows diversion of output destined for printer to a file. Binary only, source available from authors. by Alex Livshits and J-M Forgeas
Record-Replay	similar to "Journal", records and plays back mouse and keyboard events. binary only. source available from authors, Alex Livshits and J-M Forgeas

Fred Fish Disk 96

AnimPlayer	Animation reader and displayer by the combined efforts of Videoscape, Sculpt3D, Silver, Forms-In-Flight, and Animator.Appreciebe by Martin Hashetal.
Chess	Usenet posted Amiga port, non-Amiga interface. High playability. V 1.0, S. by J. Stanbeck, Amiga port by B. Leivian
Hackbench	provides source for WorkBench-Like program, for experimentation and validation of new interface ideas. Not a WBench replacement. by Bill Kinnersley
Label	Print labels with arbitrary text. V 1.3, SE available from author, M.Hansen
LineDrawer	Produces line drawings based on drawing commands stored in a text file. Includes demo that draws an outline map of the USA and state borders. V1.0, SE. by John Olsen
PopUpMenu	Example code implementing pop-up menus, reasonably compatible with Intuition menus.SE. by Derek Zahn
Tek4695	Tektronix 4695/4696 printer driver. SE. by Philip Staub
TimeRam	Fast and Chip ram test program. E. by Bruce Takahashi
WarpText	Fast text rendering routines, to be linked with application programs.Text display "as fast or faster than 'blitz'". Version 2.0 update of FF87, S by Bill Kelly

For PDS orders, Please use form on page 128

Amaze Me

Please use this order form when subscribing to Amazing Computing™, ordering Back issues, or ordering Amiga™ Public Domain Software

Name_____
Street_____
City _____ St.____ Zip _____
Amount Enclosed _____

Please circle the appropriate item: New Subscription Renewal

Please start my subscription to Amazing Computing™ with the next available issue or renew my current subscription. I have enclosed $24.00 for 12 issues in the U.S. ($30.00 Canada and Mexico, $35.00 overseas). All funds must be in U.S. Currency on a U.S. Bank

Back Issues: $4.00 each
(foreign orders add $1.00 each for Postage and Handling)

Please circle your Back issue choices below:

Vol1.1 Vol1.2 Vol1.3 Vol1.4 Vol1.5 Vol1.6 Vol1.7 Vol1.8 Vol1.9 Vol2.1 Vol2.2
Vol2.3 Vol2.4 Vol2.5 Vol2.6 Vol2.7 Vol2.8 Vol2.9 Vol2.10

Public Domain Software:
$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers

Please circle your Public Domain Software choices below:

Amicus:

A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21
A22

Fred Fish:

FF1 FF2 FF3 FF4 FF5 FF6 FF7 FF8 FF9 FF10 FF11
FF12 FF13 FF14 FF15 FF16 FF17 FF18 FF19 FF20 FF21 FF22
FF23 FF24 FF25 FF26 FF27 FF28 FF29 FF30 FF31 FF32 FF33
FF34 FF35 FF36 FF37 FF38 FF39 FF40 FF41 FF42 FF43 FF44
FF45 FF46 FF47 FF48 FF49 FF50 FF51 FF52 FF53 FF54 FF55
FF56 FFNA FF58 FF59 FF60 FF61 FF62 FF63 FF64 FF65 FF66
FF67 FF68 FF69 FF70 FF71 FF72 FF73 FF74 FF75 FF76 FF77
FF78 FF79 FFNA FF81 FF82 FF83 FF84 FF85 FF86 FF87 FFNA
FF89 FF90 FF91 FF92 FF93 FF94 FF95 FF96 FF97 FF98 FF99
FF100 FF101 FF102 (NA denotes disks removed from the collection)

Please complete this form and mail with check or money order to:

PiM Publications, Inc.
P.O.Box 869
Fall River, MA 02722

Please allow 4 to 6 weeks for delivery

Fred Fish Disk 97
Replaces FF57 for Copywrite problems

CutAndPaste	Implementaions of Unix cut and paste commands. by John Weald
GraphIt	Program to plot simple functions in 2 or 3 dimensions. by Flynn Fishman
Juggler	V1.2 of robot juggler animation. Uses HAM mode and ray tracing. by Eric Graham
MouseReader	Shareware program to read text files and view IFF files using only the mouse. by William Betz
Splines	Program to demonstrate curve fitting and rendering techniques. by Helene (Lee) Taran
Shm	simple graphics demo, approximately simulates the motion of two interacting pendulums. Includes Source by Chris Ediss

Fred Fish Disk 98

Access	16 color terminal program based on Comm V1_34. Includes Macro window, custom gadgets, colorized menus, etc. V. Beta 0.18 by Keith Young ,comm by D.J.James. E.
Backup	Writes AmigaDos disks as the backup destination. recover files from the backup disk. Requires manual decisions on disk structure. by Alan Kent SE
DCDemo	DishCat 2.3, a disk catalog program, demo limited to cataloging 100 files at a time. by Ed Arford, MicroAce Software
HdDriver	WD-1002-05 hard disk controller driver. Card capable of maintaining 3 hard disks and 4 floppies, the driver is capable of only one hard disk. by Alan Kent SED
QBase	Quick-Base, a "MailBase Management utility", define and maintain a maximum of 200 records per file. by Kevin Harrise E
Thai	Thai language quiz program. Speak or type english/Thai sentences from supplied file. by Alan Kent SE

Fred Fish Disk 99
A-Render Version .3 a Ray-Tracing Construction Set for the Amiga Computer by Brian Reed ED

Fred Fish Disk 100

Berserk	Must see animation, from Sept 87 FAUG meeting. by Leo Schwab
Conman	Console handler replacement, provides line editing and command line histories transparent to any application program that uses CON: windows. Shareware V1.0 by William Hawes. E.
WBLander	Workbench display hack game, upgrade of "Rocket" on FF65, now with sound effects. By Peter da Silva. E

Fred Fish Disk 101

CirPlane	Circular plane generator for VideoScape 3D. Generates a clockwise circular polygon with the specified number of vertibles. V1.0 by Thad Floryan SE
IconAssembler	Change Workbench Icons with IFF-brush files by Stefan Lindahl E
Microspell	Standalone spelling checker scans text files and reports errors. 1000 common word list, 43,000 word main dictionary with multiple user dictionary support. Interfaces with MicroEMACS 3.9 with an emacs macro to step through the source file, stopping at suspect words and allowing the user to option. V1.0 by Daniel Lawrence, SED
Midi	midi library and utility set. Includes Midi monitor, routing utility, status utility, and more. by Bill Barton SED
PsIntrp	Postscript Interpreter reads and previews postscript files on the screen. by Greg Lee S(assy)E
StartUps	Three C startup file replecements for standard Astartup.obj and LStarup.obj. Options include (1) BothStartup.obj, for the WorkBench programs or CLI programs with or without command line parameters. (2) WBStartup.obj, for WorkBench programs or CLI programs that require no command line parameters. (3) CLIStartup.obj for CLI programs that require command line parameters but do not need to be WorkBench runnable. by Bryce Nesbitt SE

Fred Fish Disk 102

Dbug	Machine independent macro based C debugging package. Update of FF41. by Fred Fish proffing support by Binayak Banerjee SE
Match-stuff	Heavy duty text pattern matching stuff. includes simple match text replacement capability. By Pete Goodeve
Sectorama	Recover lost or damaged data from floppy or hard disks or repair a damaged volume. by David Joiner E
SiliCon	Smart input line interpreter with window for full editing. Upgrade of FF50 by Pete Goodeve. E
Xicon	Use icons to call up scripts containing CLI commands. V2.0 upgrade of FF31. by Pete Goodeve E

To Be Continued.....

In Conclusion
To the best of our knowledge, the materials in this library are freely distributable. This means they were either publicly posted and placed in the Public Domain by their Author, or they have restrictions published in their files to which we have adhered. If you become aware of any violation of the author's wishes, please contact us by mail.

HAPPY HOLIDAYS FROM MICROILLUSIONS

THIS SEASON MICROILLUSIONS' OFFERS:

For Amiga, C64/128 and MS DOS:
FAERY TALE ADVENTURE Today's hottest game! / BLACK JACK ACADEMY / ROMANTIC ENCOUNTERS AT THE DOME.
For Amiga and soon for C64/128 and MS DOS:
LAND OF LEGENDS / PLANETARIUM / EBONSTAR / FIREPOWER / GALACTIC INVASION / TURBO.
For Amiga:
DISCOVERY and DISCOVERY EXPANSION DISKS / PHOTON VIDEO / DYNAMIC CAD / MUSIC X / DYNAMIC WORD.

microillusions™

17408 Chatsworth St., Granada Hills, CA 91344 • Inside CA 818/360-3715 • Outside CA 800/522-2041 • FAX 818/360-1464

ROME WASN'T BUILT IN A DAY, UNTIL NOW . . .

Create your own universe with SCULPT 3-D™

SCULPT 3-D brings the power of 3 dimensional solid modeling and ray tracing to the Amiga. Imagine an image: choose a color, a shape, a texture. Spin it, rotate it, extrude it into the third dimension. Pick a camera lens, set your lights, and let SCULPT 3-D create a three dimensional picture complete with shadows, reflections, and smooth shading. All in 4096 colors with true edge to edge overscan video. Easily! Automatically! Change your mind? Change the colors, textures, camera or lights in seconds and create a new image. The only limits are the boundaries of your imagination.

"I haven't had this much fun with a program since Deluxe Paint II." John Foust of Amazing Computing.

"Performance previously only available on mini and mainframe computers." Info Magazine.

Now animate your universe with ANIMATE 3-D™

Enter the fourth dimension, time. Choreograph the free flowing and simultaneous movement of objects, lights and camera through space and time. Details of object rotation, camera movements, timing and action are controlled in an easy to use graphical interface or through a simple script language. Individual objects can be linked to orchestrate complex hierarchial movements that simulate live action. Quick check wireframe playback preview's your final production: storable as a compressed animation file playable from RAM, or recorded on videotape. Additional output options include single frame VCR control or image rendering to a frame buffer card. Animations can incorporate either solid modeling or ray tracing. ANIMATE 3-D is quite simply the most powerful and easy to use animation program available for the Amiga.

Expand your universe with the BYTE BOX™

Your Amiga 500 deserves the best you can give it. More memory for more powerful applications, faster performance, better graphics, and RAM disk storage. It deserves a memory expansion system that lets you add additional memory as you need it. In easy to install and easy to afford increments. The included memory verify software provides a visual check whenever you add additional RAM. The BYTE BOX is available in a variety of configurations from 0MBytes to 2MBytes of RAM.

- Easy to install
- Fully Auto-Configure
- Fast memory that's truly fast
- Has its own power supply
- Fully tested and ready to use
- Zero wait state design
- Low profile case
- Memory check software

BYTE by BYTE™ CORPORATION

Aboretum Plaza II 9442 Capital of Texas Highway North Suite 150 Austin, TX 78759 (512) 343-4357